Adaptive Fuzzy Logic Control Applied to Socially Assistive Drones

A Case Study

Tomás Ascensão

Faculty of Aerospace Engineering

TU Delft

Adaptive Fuzzy Logic Control Applied to Socially Assistive Drones

A Case Study

by

Tomás Ascensão

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday November 1, 2021 at 14:30 AM.

Student number:	5126312	
Project duration:	September 3, 2020 – November 1, 2021	
Thesis committee:	Dr. ir. M.M. van Paassen,	TU Delft, chair
	Dr. A. Jamshidnejad,	TU Delft, supervisor
	Dr. A.A. Nunez Vicencio ,	TU Delft, external examiner

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

The following thesis marks the end of a two-year chapter of my life, filled with both professional and personal growth. Needless to say, I could not have made it this far on my own and thus I would like to express my gratitude to those who supported me along my journey to become a full-fledged aerospace engineer (for now...).

Firstly, I would like to thank Dr. Anahita Jamshidnejad for her guidance over the last year. Your constructive feedback and meticulous eye to detail and have proven to be second to none and have greatly shaped the quality of the outcome of this research. I would also like to thank my friends from Delft, with whom I gladly shared the last two years and from whom I learned so much throughout this period. In particular, and as promised, I would like to express my gratitude to my housemates. This past year would not have been nearly as enjoyable without your daily presence, which never failed to liven up my mood, even in the middle of a global pandemic. Thank you to all my friends and family in Portugal for your continuous and relentless support. Last but most certainly not least, I would like to thank my mother, to whom I owe everything that I have achieved thus far.

Tomás Ascensão
Delft, September 2021

Contents

List of Figures

List of Tables

List of Acronyms

ABC Artificial Bee Colony.

ANN Artificial Neural Networks.

ASD Autism Spectrum Disorder.

COG Center of Gravity.

DMT Dance Movement Therapy.

DRL Deep Reinforcement Learning.

FIS Fuzzy Inference System.

FLAC Fuzzy Logic Adaptive Controller.

FLC Fuzzy Logic Controller.

GA Genetic Algorithms.

GUI Graphical User Interface.

IPU Image Processing Unit.

KBC Knowledge Based Controller.

MIMO Multiple Input Multiple Output.

ML Machine Learning.

MOM Mean of Maxima.

MRAS Model Reference Adaptive System.

PSO Particle Swarm Optimization.

QEN Q Estimator Network.

RL Reinforcement Learning.

SAD Socially Assistive Drone.

SAR Socially Assistive Robot.

SIR Social Interactive Robotics.

STC Self Tuning Controller.

TD Temporal Difference.

TS Takagi-Sugeno.

TSK Takag-Sugeno-Kang.

UAV Unmanned Aerial Vehicle.

VR Virtual Reality.

Nomenclature

α	Learning rate
$\dot{x}$	Time derivative of x
$\hat{Q}(s,a)$	Estimate of the action-value function
$\hat{V}(s)$	Estimate of the state-value function
λ	Discount factor
$\mathbb{E}[\ldots]$	Expected value operator
$\mathbb{R}$	Real number set
$\mathcal{R}$	Fuzzy Rule Set
$\mu_A(x)$	Membership referent to fuzzy set A
∇	Gradient operator
$\pi(s)$	Policy function
$\pi^*(s)$	Optimal Policy
σ	Standard deviation
$\vec{w}$	Weight vector
$\vec{x}$	State vector
A	Set of actions
a	Action
c	Mean
e	Error
G	Cumulative Reward
$Q^*(s,a)$	Optimal action-value function
$Q_\pi(s,a)$	Action-value function
R	Fuzzy Rule
$r(s,a)$	Reward Function
r_t	Reward at time step t
S	Set of states
T	Final time step
t	Time
u	Control input
$V^*(s)$	Optimal state-value function
$V_\pi(s)$	State-value function
x_{ref}	Reference for x

I

Scientific Article

Noname manuscript No.
(will be inserted by the editor)

Socially Assistive Drones Performing Dance Movement Therapy

An Adaptive Fuzzy-Logic-Based Control Approach

Tomás Ascensão · Anahita Jamshidnejad

the date of receipt and acceptance should be inserted later

Abstract
With the number of diagnosed cases rising every year, Autism Spectrum Disorder (ASD) is in need of novel therapeutic approaches to counteract the social and motor impairments inflicted by it. In this research, one of such therapeutic approaches, movement therapy (in particular Dance Movement Therapy (DMT)), is combined with the concept of Socially Assistive Robots (SARs) to assess the viability of employing a new type of SAR. This SAR consists in the first ever Socially Assistive Drone (SAD): a quadrotor drone of reduced size conceived for maintaining human-drone interaction during therapeutic sessions meant for children diagnosed with ASD. The main focus of this paper consists in developing an adaptive control system based on Fuzzy Logic Control due to its inherent advantages such as intuitiveness or the ease with which expert knowledge can be introduced into control systems. In total, four different behavioural modes are developed for the SAD, together with an adaptive algorithm which influences both a set of adjustable parameters associated with each mode and the likelihood of it being selected based on the user specific preferences. The four behavioural modes and their respective adaptive algorithms are tested on 10 participants in 30-minute sessions. The variations in the adaptive parameters and the likelihood of each mode being selected for each individual are registered and reveal that the SAD is able to adapt its behaviour to each participant based on their respective levels of engagement.

Keywords Adaptive Control · Fuzzy Logic Control · Socially Assistive Robots · Drone · Autism Spectrum Disorder · Dance Movement Therapy

Nomenclature

$\bar{e}_{\text{ep}}$	average episodic engagement
$\bar{p}_{\text{ep}}$	average episodic performance
Δx_{hand}	hand distance to vertical center line
Δx_{user}	user distance to vertical center line
Δy_{hand}	hand distance to horizontal center line
$\rho_{\text{SAD}}^{\text{GMi}}$	i^{th} game mode lateral output
$\zeta_{\text{SAD}}^{\text{GMi}}$	i^{th} game mode vertical output
$h_{\text{SAD}}^{\text{def}}$	default altitude
h_{SAD}	SAD altitude
h_{user}	user height
$\upsilon_{\text{SAD}}^{\text{standby}}$	yaw rate output of the standby controller
$\tilde{r}_t$	register containing anomalies
σ	standard deviation
τ_{c}	centered threshold
τ_{m}	motion threshold
t_{f}	time of failure
$\tilde{C}$	anomaly coefficient
$\tilde{x}_t$	x coordinate of register with anomaly
$\tilde{y}_t$	y coordinate of register with anomaly
$\dot{x}_{\text{user}}^{\text{chest}}$	user chest horizontal velocity
$\dot{x}_{\text{user}}^{\text{wrist}}$	user wrist horizontal velocity
$\dot{y}_{\text{user}}^{\text{chest}}$	user chest vertical velocity
$\dot{y}_{\text{user}}^{\text{wrist}}$	user wrist vertical velocity
x_{image}	number of pixels in horizontal direction
x_{user}	user horizontal position
y_{image}	number of pixels in vertical direction
$\zeta_{\text{SAD}}^{\text{standby}}$	vertical output of the standby controller
$\pi_{\text{SAD}}^{\text{standby}}$	pitch rate output of the standby controller
A	area
a	slope
b	y-intercept
K	gain
N_{trial}	number of trials
r	position register
t_i	time instant in the i^{th} measurement
T_p	episodic average pause time
t_{w}	wait time
w	personalization weight
x	horizontal coordinate in captured image
x_{user}	user horizontal position
y	vertical coordinate in captured image
y_{user}	user vertical position

1 Introduction

Autism Spectrum Disorder (ASD) refers to a range of complex neuro-developmental conditions that result in difficulties in social interactions and verbal and non-verbal communication. The prevalence of ASD is around 1.7% (2.7% of boys and 0.67% of girls are affected by ASD). According to Ricks et al. [1], almost all people with ASD exhibit some level of language impairment, with a large percentage of them being completely non-verbal. Moreover, according to Kopp et al. [2], 80% - 90% of people with ASD are affected by an impaired development of motor skills[1]. Several studies (see, e.g., [3; 4; 5; 6; 7]) show that the severity of motor skill impairments in people with ASD is directly correlated with the severity of their social and communication impairments.

In recent years, with the increased number of diagnosed cases, ASD has become a core topic of research in various related disciplines. Although there is currently no cure for ASD, people with ASD can improve their symptoms and quality of life via early, long-term therapeutic interventions. Movement therapy, or more specifically Dance Movement Therapy (DMT), has gained lots of interest among researchers since its advent in the 1970s [8], as it can significantly contribute to the improvement of motor skills and well-being of both verbal and non-verbal children with ASD, who may not benefit significantly from other therapeutic interventions [9]. DMT for ASD therapy includes mirroring (i.e., matching, reflecting, or echoing) the movements of the therapist by the client (with the aim of improving empathetic expressions) and encouraging the client to initiate more spontaneous movements and mirroring them by the therapist (with the aim of improving the connection with and the exploration of the environment in clients).

The main challenges for implementing DMT for people with ASD include (1) the need for personalizing the DMT plans based on the stage of ASD and specific needs of every client, (2) making the intervention available - with any desired frequency - and engaging in long term for all clients, and (3) keeping the corresponding costs low, so that the intervention is affordable for a larger group of people. Therefore, seeking alternative approaches to provide personalized, affordable DMT for all clients.

Socially Assistive Robots (SARs), introduced by Ricks et al. [1], are assistive (i.e., they provide aid or support for humans, in rehabilitation, education, mobility, etc.) and socially interactive (i.e., they maintain social interactions with humans) robots. Several researches (see, e.g., [10; 11; 12; 13; 14; 15]) have shown that therapeutic interventions via SARs for people with ASD can significantly improve the impact and therapeutic outcome of these interventions. Correspondingly, various SARs have been developed or adapted for use in ASD therapy sessions. Examples include the seal robot Paro [16], the humanoid robot NAO [17], and the mascot Charlie [18] shown in Figure 1.

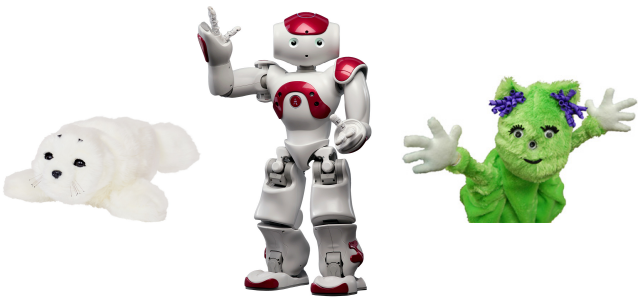


Fig. 1 Different SARs used for ASD therapy in literature (from left-hand side: Paro [16], Nao [17], and Charlie [18])

In order to achieve optimal therapeutic performance and outcome via SARs, these robots should possess the following key characteristics:

1. **Desirable appearance:** Although humanoid robots may generally have the highest acceptance by humans, for people (especially children) with ASD, the use of humanoid robots may hinder the effectiveness of interactions. This is mainly because people with ASD tend to experience discomfort around humans (and similarly around robots that resemble humans) and thus avoid interacting with them [1]. Briefly speaking, the less complex and expressive the appearance of a robot, the more desirable the robot for ASD therapeutic applications.
2. **Adaptivity:** For effective interactions with humans, SARs should adapt their behaviour and decisions according to the environmental and behavioral changes of the client. The adaptivity of SARs can positively affect their acceptance by humans, and can encourage humans to interact with them in longer terms [19].
3. **User-friendliness:** Therapists, clinicians, and caregivers should be able to operate or re-program SARs easily and without need for in-depth technical knowledge. User-friendliness is mainly important for the user interface (e.g., a GUI) of a SAR, i.e., the interface should be simple and intuitive in providing the desired commands and in entering or accessing data of therapy sessions.
4. **Responsiveness:** Since people with ASD may exhibit unpredictable patterns of behavior and interaction or may tend to withdraw from the interactions from time to time [20], it is important for a SAR to be responsive to the immediate changes a therapist or caregiver may provide (to improve the interactions) during a therapy session. Note that while adaptivity is related to the autonomous performance of SARs, responsiveness is related to the

[1] Motor skills include abilities - learned by humans - that can result in predetermined movement outcomes with maximum certainties.

robot's performance with regards to the external control inputs that it receives.

5. **Autonomy:** Some levels of autonomy is required for SARs in order to continue performing independently without constant supervision of therapists or caregivers. This is particularly important, since it allows therapists or caregivers to focus on other therapeutic aspects that are not necessarily addressed by the SAR or to contribute to several parallel therapy sessions.

Considering the five characteristics mentioned above, drones are highly suitable candidates for assisting in therapeutic interventions for ASD. However, drones have never been used in systematic therapeutic interventions for ASD and more generally as SARs. High mobility and manoeuvrability and the three degrees of freedom of drones, which are fundamental for performing DMT, as well as their simple non-humanoid appearance, which makes people with ASD more comfortable and interested in getting involved in interactions with drones[2], makes them promising candidates for ASD therapy. Therefore, in this paper we introduce a new concept, socially assistive drones or briefly SADs, and we develop the first ASD used for performing personalized DMT in live interactions with humans.

Adaptivity, user-friendliness, responsiveness, and autonomy of a SAD are mainly related to the approaches that will be developed to steer SADs. More specifically, adaptivity and autonomy can be provided by proper development of the SAD's control system, while user-friendliness and responsiveness are linked to both the control system and the user interface of a SAD. The main focus of this paper is on the development of the control system of a SAD, where adaptive fuzzy-logic-based approaches are considered.

Fuzzy-logic-based controllers or FL controllers were first introduced by Mamdani [21] (based on the concept of fuzzy sets [22]) and were later on extended by Takagi and Sugeno [23]. FL controllers make decisions using a fuzzy inference system and according to a rule base that consists of rules, which are formulated as if-then statements that include linguistic terms. The input and output of an FL controller are fuzzy values. Therefore, a fuzzifier and a defuzzifier are used to transform the crisp values into fuzzy ones and vice versa, since sensors and actuators of the controlled system usually perform according to crisp values. FL controllers have the following advantages: FL controllers are usually model-free, which makes suitable in cases where no model of the controlled system is available. Moreover, FL controllers perform based on decision making approaches that are very close to decision making approaches of humans. When building up the rule base of an FL controller, human expert knowledge (expressed via linguistic terms) can directly be incorporated within the controller. In Takagi-Sugeno-Kang (TSK) FL controllers [23], although the antecedent of the rules include fuzzy sets, their consequent directly produces crisp values based on an affine combination of the inputs. In this paper, TSK-based FL controllers are used for developing the controllers of the SAD, since the TSK approach is computationally efficient, is easy to be tuned, and cab easily be made adaptive.

[2] Children with ASD may feel intimidated by the humanoid appearance of robots [1]. In contrast, the experiences of drone summer camps in US for children with ASD have shown that drones are very appealing to these children (see, for instance, https://wisconsinlife.org/story/group-works-with-autistic-children-to-build-fly-small-drones/ and https://trajectorymagazine.com/tatts-cultivating-social-and-employment-skills-with-drones/).

2 Main Contributions & Structure of the Paper

In this paper we develop, implement, and assess an adaptive fuzzy-logic-based control system for a first ever socially assistive drone or SAD, introduced to perform DMT sessions for people with ASD. In addition to the key characteristics of adaptivity, user-friendliness, responsiveness, and autonomy, the developed control system provides excellent potentials for personalization of the SAD, which is a crucial requirement in ASD therapies. In particular, the main contributions of this paper include:

1. An analysis on the first experiments ever conducted on a SAD interacting with human subjects
2. An assessment of the performance of the controller developed, particularly in terms of adaptivity
3. An evaluation of the viability of SAD development under the present testing conditions, including the control algorithm employed and ANN-based IPU usage in real-time

In particular, the control system and its underlying algorithms are discussed in detail, together with a brief overview of the IPU used. Lastly, the Socially Assistive Drone developed is tested on 10 test subjects and the obtained results such as overall performance, and adaptivity algorithm performance are presented and discussed.

The rest of the paper is organized as it follows: Section 3 addresses the proposed methodologies in detail. This includes a description of the Human-SAD interaction requirements and the data capturing procedure, in particular, the two artificial neural networks comprised in the image processing unit. Furthermore, this section addresses the data analysis module, along with its inherent motion processing algorithm and the therapeutic scenarios developed, referred to as game modes, followed by the implemented adaptivity and personalization algorithms. Subsequently, Section 4 refers to the particular case study presented and, as such, addresses the equipment used, as well as the experimental setup and conditions available. Moreover,

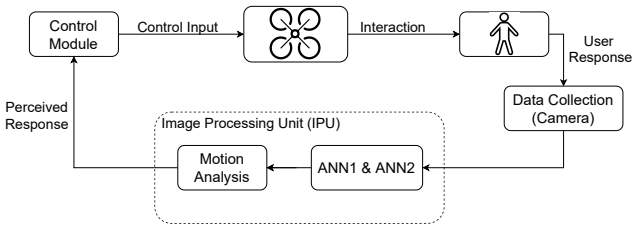


Fig. 2 Schematic representation of the human-SAD interaction procedures

it entails the case-specific implementation of all the components introduced in the previous sections, including the image processing unit, the motion processing algorithm, all the game modes developed and the adaptivity and personalization modules. Section 5 presents and discusses the results obtained in the experimental procedure conducted. This is followed by a discussion of limitations faced and possible topics for future work, in Section 6. Lastly, in Section 7, the main conclusions obtained from the experimental procedure are presented.

3 Proposed Research Methodologies

In this section, we explain data analysis and control approaches developed for the SAD, which together build up the decision making module that steers the SAD to perform DMT plans autonomously and in a personalized way in live interactive sessions with humans.

3.1 Human-SAD Interactions

In order for the SAD to autonomously perform systematic DMT plans and to maintain the interactions with human users in long term, the SAD needs to successfully accomplish the following tasks:

1. To gather relevant data from users and to process them and generate information that is reliable and comprehensive for the assessments and analyses.
2. To analyze the generated information - according to the assessment criteria and purposes of the interactions - and to make assessments and perceptions.
3. To inject the results of the assessments and perceptions into the SAD's control system (i.e., the decision-making module of the SAD), which generates control inputs that steer the SAD's actions according to the aims of the interactions.

Figure 2 illustrates the three steps indicated above. In the next sections, we provide details on these steps.

3.2 Data Capturing via SAD's Camera

From now on, we mean a quadcopter equipped with a camera whenever we refer to a SAD. The camera

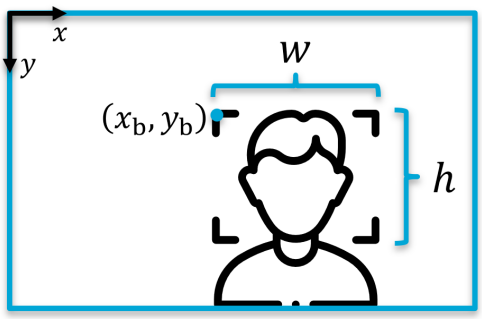


Fig. 3 Output of ANN 1: Parameters regarding the face-box that surrounds the user's face, including the coordinates of the top left corner of the face-box (x_b, y_b), its width w, and height h

enables the SAD to record live video footage of its surroundings during DMT sessions. We assume that at most one human at a time is present in the frame of view of the SAD. A fixed time interval, referred to as the *wait time*, is considered for the SAD. During the wait time the SAD remains in the same position and records all the relevant positions of the user (details are given in Sections 3.2.1 and 3.2.2).

In order for the SAD to identify the presence of a user within the recorded video footage and to follow and evaluate the movements of the user, two artificial neural networks (ANNs) and a motion processing algorithm are developed, which together form the image processing unit (IPU) and run on a remote computer and process the corresponding images in real time (See Figure 2). In this section we discuss the details of the two ANNs that are used in the data analysis module of the SAD.

3.2.1 ANN 1: Face Detection Module

The first ANN, referred to as ANN 1, provides information for the SAD to identify the face of a human within the captured images by sketching a rectangle (called the face-box) that delimits the user's face (see Figure 3). More specifically, the output of ANN 1 includes the following values: (1) coordinates (x_b, y_b) of the upper left corner of the face-box, (2) the width, w, of the face-box, and (3) the height, h, of the face-box. These values are all given in pixels. Note that the largest blue rectangle in Figure 3 shows the SAD's frame of view.

3.2.2 ANN 2: Body/Joint Detection Module

The second ANN, referred to as ANN 2, is developed by Openpose Python [24; 25], a real-time detection library that detects body, hand, face, and foot keypoints of a user in a single image. ANN 2 is responsible for identifying the joint positions of a user that has been detected. Overall, a total of 18 joint positions (see Figure 4) can be identified by ANN 2. Similarly to ANN 1, each pair of coordinates for these joints is expressed in pixels within the picture reference frame.

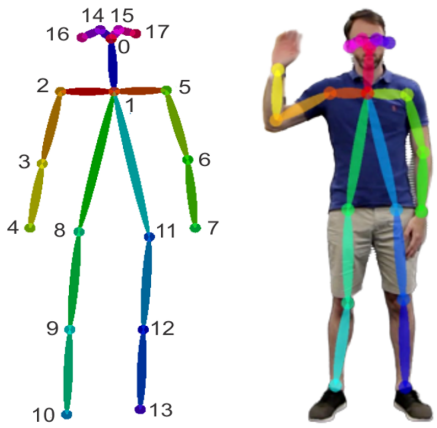


Fig. 4 Output of ANN 2: Coordinates of the 18 joints of the user's body illustrated in the right-hand side plot

Remark 1 In addition to the coordinates and values generated by ANN 1 and ANN 2, the observation time associated with each of these values is also stored on the remote computer. This information will later on be used by the SAD to estimate the speed of the user's movements, which is crucial both for assessment of the interactions and for determination of the speed and frequency of the various DMT plans that will be proposed by the SAD.

3.3 Data Analysis Module

The outputs of the ANNs are injected as input into the motion processing algorithm of the SAD, which uses these inputs to (1) estimate the current position of the user, (2) evaluate the level of engagement and performance of the user according to the DMT plan, and (3) capture the movements that are initiated by the user in case the SAD is expected to follow or mirror these movements

Next, we explain the motion processing algorithm in detail.

3.3.1 Motion Processing Algorithm

The information regarding every captured measurement captured by either of the ANNs is skimmed to contain three values: (1) t_i, the time instant (in second) when the i^{th} measurement was captured, (2) x_{t_i}, the horizontal coordinate (in pixels) of the relevant joint for the i^{th} measurement, and (3) y_{t_i}, the vertical coordinate (in pixels) of the relevant joint for the i^{th} measurement. For the sake of simplicity of the notations we use $r_{t_i} = (x_{t_i}, y_{t_i})$. The available data is then registered into five categories of motions: motion to the left, motion to the right, motion upwards, motion downwards, and pause. To identify these motions, every two consecutive coordinates r_{t_i} and $r_{t_{i+1}}$ are considered:

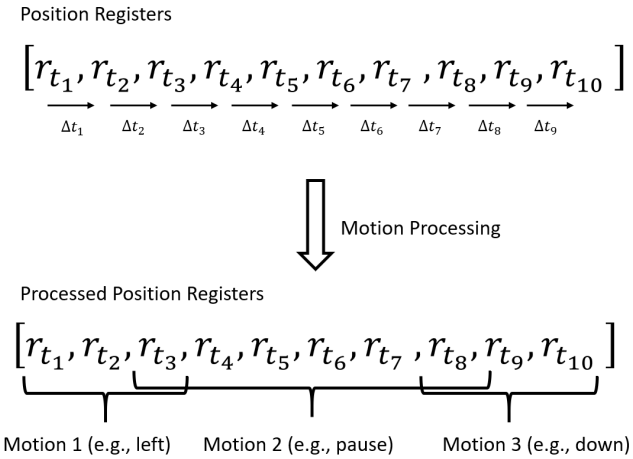


Fig. 5 Motion processing algorithm generating some motion categories (specified by Motion 1, Motion 2, and Motion 3)

1. In case the two coordinates are (almost) overlapping, i.e., $|r_{t_i} - r_{t_{i+1}}| < \tau_{\text{m}}$, with τ_{m} a motion threshold, then the coordinates r_{t_i} and $r_{t_{i+1}}$ belong to the category "pause".
2. In case the distance between the two consecutive coordinates is larger than or equal to the motion threshold, i.e., $|r_{t_i} - r_{t_{i+1}}| \geq \tau_{\text{m}}$, then a motion is detected. The motion is horizontal when $|y_{t_i} - y_{t_{i+1}}| < \gamma |x_{t_i} - x_{t_{i+1}}|$, and is vertical when $|x_{t_i} - x_{t_{i+1}}| < \gamma |y_{t_i} - y_{t_{i+1}}|$, with $0 < \gamma < 1$ a ratio that can be identified per person[3].
3. To specify the heading of the motion, i.e., left or right for horizontal motions and downwards or upwards for vertical motions, the sign of, respectively, $x_{t_i} - x_{t_{i+1}}$ and $y_{t_i} - y_{t_{i+1}}$ is considered. A positive sign indicates a motion to the right or downwards, while a negative sign indicates a motion to the left or upwards.

Figure 5 illustrates the motion categorization for a sample of 10 captured coordinates. Finally, considering the coordinates and registered time instants, a speed (in pixels per second) is associated to every identified motion category.

Fault tolerance for motion processing algorithm:
For the motion processing algorithm to possess an acceptable level of fault tolerance, it should cope with (1) *missing data*, i.e., when the user and/or the relevant joints are not identified by the IPU, as well as with (2) *erroneous data*, i.e., when the position of the user and/or the corresponding joints have been captured, but involve (non-negligible) errors. A missing coordinate or erroneous coordinate may disturb the

[3] In reality, it is very unlikely for a user to move consistently in exactly either a horizontal or a vertical direction. For instance, a user who aims to move horizontally may still make some small vertical movements inadvertently. The ratio γ has been considered in analysis of the user's motions to determine whether or not these secondary motions should be considered by the SAD.

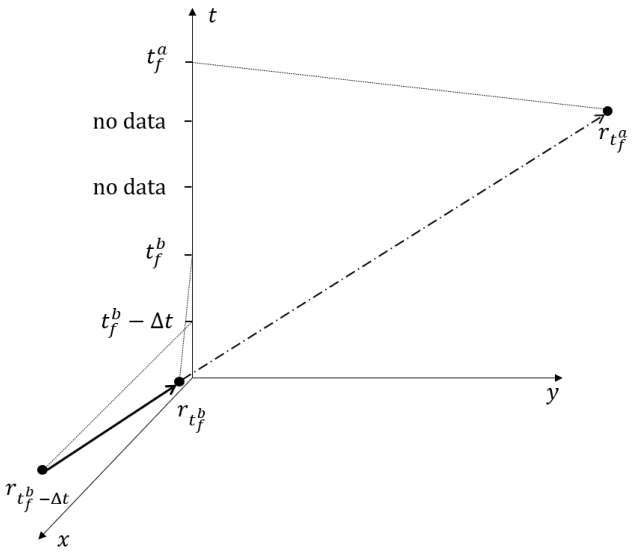


Fig. 6 Missing data: Uniform movement during the time interval when data has not been captured

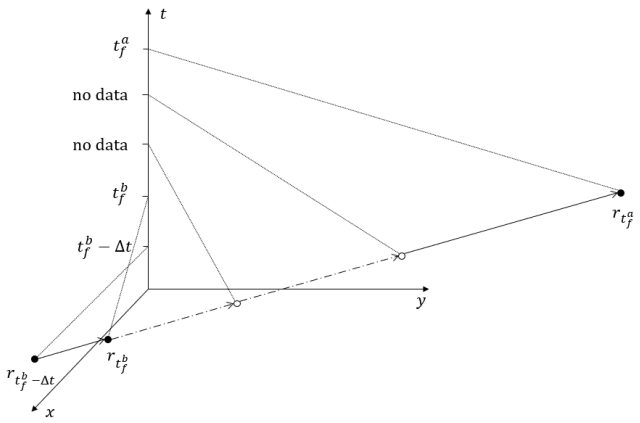


Fig. 7 Missing data: Movement in the same direction with a varying speed during the time interval when data has not been captured

performance of the motion processing algorithm, and thus should be replaced by a plausible option.

A missing coordinate may occur because the IPU fails to capture any coordinates or because the user leaves the frame of view of the SAD. In case the number of consecutive missing coordinates is small (e.g., not larger than 2), the chance that the user has left the frame of image and suddenly has returned is negligible. Alternatively, in case the number of consecutive missing coordinates is considerable (e.g., larger than 2), the chance that the user has left the frame of the image is high.

Missing coordinates due to IPU failure: The first time instant that corresponds to a missing coordinate is called the time of failure and is denoted by t_f. The most recent and the first next time instants with respect to t_f when a reliable measurement has been captured are specified by t_f^b and t_f^a. The coordinates corresponding to time instants t_f^b and t_f^a are represented by $r_{t_\mathrm{f}^\mathrm{b}}$ and $r_{t_\mathrm{f}^\mathrm{a}}$. In case the coordinates $r_{t_\mathrm{f}^\mathrm{b}}$ and $r_{t_\mathrm{f}^\mathrm{a}}$ imply the same category of the motion as the most recent time interval $\left[t_\mathrm{f}^\mathrm{b} - \Delta t, t_\mathrm{f}^\mathrm{b}\right]$ (with Δt the fixed time step for capturing a measurement by the IPU), and the speed for moving from coordinate $r_{t_\mathrm{f}^\mathrm{b}}$ to coordinate $r_{t_\mathrm{f}^\mathrm{a}}$ within time interval $\left[t_\mathrm{f}^\mathrm{b}, t_\mathrm{f}^\mathrm{a}\right]$ compared to the speed of the captured movement within time interval $\left[t_\mathrm{f}^\mathrm{b} - \Delta t, t_\mathrm{f}^\mathrm{b}\right]$ indicates a uniform motion (see Figure 6), then the motion processing algorithm ignores the missing intermediate points (as they are assumed to be positioned uniformly between $r_{t_\mathrm{f}^\mathrm{b}}$ and $r_{t_\mathrm{f}^\mathrm{a}}$).

In case based on the coordinates $r_{t_\mathrm{f}^\mathrm{b} - \Delta t}$ and $r_{t_\mathrm{f}^\mathrm{b}}$, and $r_{t_\mathrm{f}^\mathrm{b}}$ and $r_{t_\mathrm{f}^\mathrm{a}}$ the direction of the motion has remained the same within time intervals $\left[t_\mathrm{f}^\mathrm{b} - \Delta t, t_\mathrm{f}^\mathrm{b}\right]$ and $\left[t_\mathrm{f}^\mathrm{b}, t_\mathrm{f}^\mathrm{a}\right]$, while the speed of movement has changed (see Figure 7), then n intermediate coordinates, with $n = (t_\mathrm{f}^\mathrm{a} - t_\mathrm{f}^\mathrm{b})/\Delta t - 1$, between the coordinates $r_{t_\mathrm{f}^\mathrm{b}}$ and $r_{t_\mathrm{f}^\mathrm{a}}$ are specified by the motion processing algorithm, such that the magnitude of the speed corresponding to these intermediate time intervals evolves according to a fixed rate and the SAD's position for time step t_f^a reaches the position captured for this time step by the IPU. More specifically, the rate of changes of the speed is given by:

$$\Delta v = \frac{2\left(|r_{t_\mathrm{f}^\mathrm{a}} - r_{t_\mathrm{f}^\mathrm{b}}| - |r_{t_\mathrm{f}^\mathrm{b}} - r_{t_\mathrm{f}^\mathrm{b} - \Delta t}|\right)}{n(n+1)\Delta t} \tag{1}$$

When according to the motion processing algorithm, the direction of the captured motion has changed between t_f^b and t_f^a (see Figure 8), the intersection of the piece of line that connects the points corresponding to $r_{t_\mathrm{f}^\mathrm{b} - \Delta t}$ and $r_{t_\mathrm{f}^\mathrm{b}}$ with the piece of line that connects the points corresponding to $r_{t_\mathrm{f}^\mathrm{a}}$ and $r_{t_\mathrm{f}^\mathrm{a} + \Delta t}$ will be considered as the point where the direction of the motion has changed. In case based on the most recent and the most posterior time intervals from the time interval when no data has been captured, the speed of the motion has also changed the path determined by the motion processing algorithm from time instant t_f^b to time instant t_f^a will be divided into n intermediate coordinates, such that the magnitude of the speed corresponding to these intermediate time intervals evolves according to a fixed rate as explained before (in this case the SAD will change the direction of the motion according to the determined turning point, but the magnitude of the speed will follow the approach that has been explained earlier on).

Missing coordinates due to the user leaving the SAD's frame of view: When the coordinates corresponding to the position of the user has not been registered for a large number of time steps, the motion processing algorithm assumes that the user has left the frame of view of the SAD. Correspondingly, the algorithm considers the last registered coordinates $\left(x_\mathrm{user}^\mathrm{L}, y_\mathrm{user}^\mathrm{L}\right)$ of the user before they leave the SAD's frame of view and determines the distance d_i^L of this position with respect to the eight border points b_i with $i \in \{1, \ldots, 8\}$ (see Figure 9 and 10) of the SAD's frame of view.

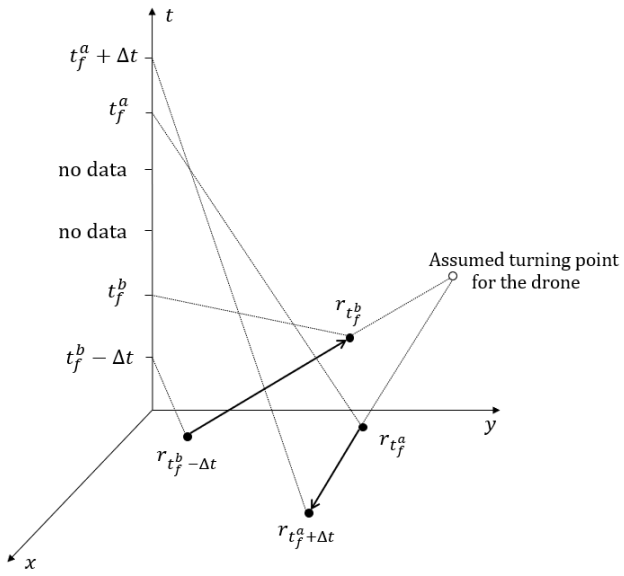


Fig. 8 Missing data: Direction of movement has changed during the time interval when data has not been captured

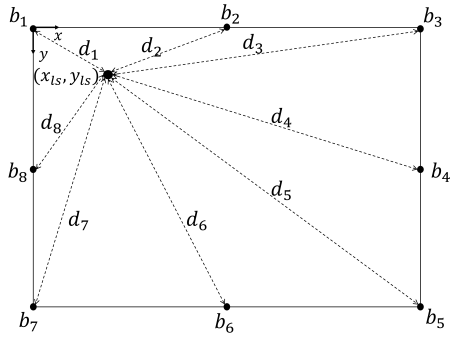


Fig. 9 Missing data due to the user leaving the SAD's frame of view: A case where no new position has been registered for the user

In case there is no later registered coordinates for the user (see Figure 9), the algorithm considers the border point that is the closest to the point $\left(x^{\mathrm{L}}_{\mathrm{user}}, y^{\mathrm{L}}_{\mathrm{user}}\right)$ in order to fill in the missing coordinates. However, when there is a new registered position $\left(x^{\mathrm{R}}_{\mathrm{user}}, y^{\mathrm{R}}_{\mathrm{user}}\right)$ for the user, the motion processing algorithm considers this point as the return position of the user (see Figure 10). Correspondingly, the distances d_i^{R} of this return position with respect to the eight border points as well as the distances $d_j^{\mathrm{O,R}}$ and $d_k^{\mathrm{O,L}}$ between the positions corresponding to $\left(x^{\mathrm{O},j,k}_{\mathrm{user}}, y^{\mathrm{O},j,k}_{\mathrm{user}}\right)$ are computed, where $j, k \in \{1, \ldots, 8\}$. Note that $\left(x^{\mathrm{O},j,k}_{\mathrm{user}}, y^{\mathrm{O,j,k}}_{\mathrm{user}}\right)$ corresponds to the intersection point of the lines that pass through the point corresponding to $\left(x^{\mathrm{L}}_{\mathrm{user}}, y^{\mathrm{L}}_{\mathrm{user}}\right)$ and the border point b_j, and the point corresponding to $\left(x^{\mathrm{R}}_{\mathrm{user}}, y^{\mathrm{R}}_{\mathrm{user}}\right)$ and the border point b_k. Finally, the coordinates of the two border points b_j and b_k, for which $d_j^{\mathrm{L}} + d_j^{\mathrm{O,L}} + d_k^{\mathrm{R}} + d_k^{\mathrm{O,R}}$ is the smallest are used to fill in the missing registrations (in Figure 10, e.g., $b_j = b_2$ and $b_k = b_4$).

Erroneous data: An erroneous is identified by abnormal deviations from a specific motion pattern. Mathematically speaking, an error corresponds to a registered position $\tilde{r}_t$ that satisfies both of the following conditions:

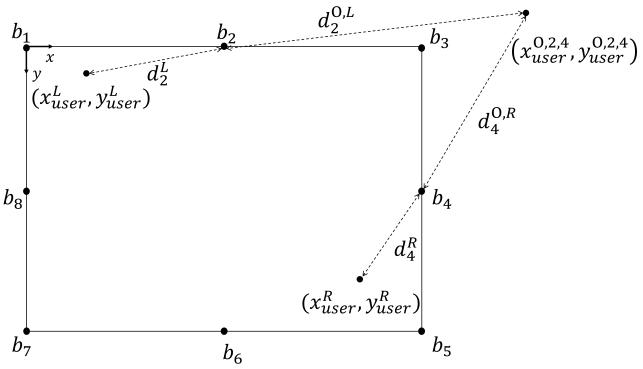


Fig. 10 Missing data due to the user leaving the SAD's frame of view: A case where the user has left the SAD's frame of view, and later on a new position has been registered for the user

$$\min\left\{||\tilde{r}_t - r_{t-\Delta t}||, ||\tilde{r}_t - r_{t+\Delta t}||\right\} > \tau_{\mathrm{m}} \tag{2}$$

$$\max\left\{||\tilde{r}_t - r_{t-\Delta t}||, ||\tilde{r}_t - r_{t+\Delta t}||\right\} > \tilde{C}||r_{t+\Delta t} - r_{t-\Delta t}|| \tag{3}$$

where $\tilde{C} > 1$ is called the anomaly coefficient and τ_{m} is the motion threshold defined in Section 3.3.1. In order to detect erroneous data, the motion processing algorithm should first assure that a motion across the registered positions $r_{t-\Delta t}$, $\tilde{r}_t$, and $r_{t+\Delta t}$ has been detected, which is taken care of via condition (2). Condition (3) defines a limit on the deviation from a given motion pattern for every registered position. Erroneous data at time instant t is identified whenever the distance between the associated registered position $\tilde{r}_t$, and either of the previous, i.e., $r_{t-\Delta t}$, or following, i.e., $r_{t+\Delta t}$, registered positions is larger than $\tilde{C}$ times the distance between the previous and the following registered positions.

Figure 11 shows two examples with absence and presence of erroneous data. In this figure, satisfaction of condition (2) means that d_{23} (which is assumed to be smaller than d_{12}) is larger than τ_{m}, meaning that r_{t_2} falls outside of the illustrated circular areas of radius τ_{m}. The left-hand side plot shows a case where condition (3) (supposing that $\tilde{C} = 2$) does not hold for the registered position r_{t_2}, i.e., d_{12} (which is supposed to be larger than d_{23}) is not larger than $2d_{13}$. Therefore, there is no erroneous data detected for time instant t_2. However, in the right-hand side plot, d_{12}, for example, is larger than $2d_{13}$, which according to (3) implies erroneous data for r_{t_2}.

Whenever erroneous data is identified, the corresponding registered position should be corrected. Suppose that we have either of the following cases:

$$|x_{t+\Delta t} - x_{t-\Delta t}| < 2\tau_{\mathrm{m}} \tag{4}$$

$$|y_{t+\Delta t} - y_{t-\Delta t}| < 2\tau_{\mathrm{m}} \tag{5}$$

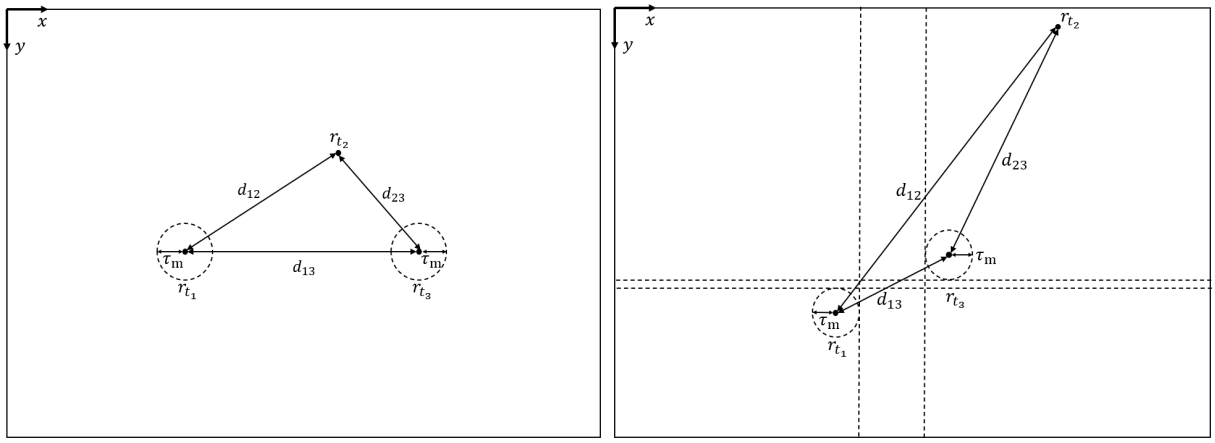


Fig. 11 No erroneous data for r_{t_2} (left-hand side picture). Erroneous data detected for r_{t_2} (right-hand side picture)

Then the registered position that is prone to error is ignored and the SAD considers $r_{t-\Delta t}$ and $r_{t+\Delta t}$ as two consecutive registered positions.
In case (4) does not hold, while the following condition holds (see the sub-area in between the dashed vertical lines in the right-hand side plot in Figure 11):

$$\min\Big\{x_{t-\Delta t}, x_{t+\Delta t}\Big\} + \tau_{\mathrm{m}} < \tilde{x}_t < \max\Big\{x_{t-\Delta t}, x_{t+\Delta t}\Big\} - \tau_{\mathrm{m}}, \quad (6)$$

only the y-coordinate of $\tilde{r}_t$ should be corrected, i.e.:

$$x_t = \tilde{x}_t \quad (7)$$

$$y_t = 0.5\Big(y_{t-\Delta t} + y_{t+\Delta t}\Big) \quad (8)$$

If (4) does not hold, while the following condition holds (see the sub-area in between the dashed horizontal lines in the right-hand side plot in Figure 11):

$$\min\Big\{y_{t-\Delta t}, y_{t+\Delta t}\Big\} + \tau_{\mathrm{m}} < \tilde{y}_t < \max\Big\{y_{t-\Delta t}, y_{t+\Delta t}\Big\} - \tau_{\mathrm{m}}, \quad (9)$$

only the x-coordinate of $\tilde{r}_t$ is corrected. We have:

$$x_t = 0.5\Big(x_{t-\Delta t} + x_{t+\Delta t}\Big) \quad (10)$$

$$y_t = \tilde{y}_t \quad (11)$$

3.4 SAD's Control System: DMT Game Modes

The main goal of the SAD is to sustain effective interactions according to systematic DMT plans with a user in long term (e.g., during a DMT session of 30 min for this research). For a systematic autonomous planning of DMT, we have considered a number of therapeutic scenarios, which we refer to as "game modes". The proposed game modes generally fall within one of the following two categories:

1. *Passive game modes*: These game modes mainly aim at promoting empathetic illustrations. The SAD mirrors the movements of the user (i.e., the SAD takes a passive role in the DMT interactions).
2. *Active game modes*: These game modes mainly aim at enhancing the connection of the user with their environment, thus the user is expected to follow the harmonic movements that are initiated by the SAD (i.e., the SAD takes an active role in the DMT interactions).

In passive game modes the wait time is given to the SAD to capture the motions of the user, which the SAD will afterwards mirror, while in active game modes the wait time is considered after the SAD performs a specific DMT movement and should assess the reactions or responses of the user.

Overall, four game modes in both active and passive categories are proposed in this research, which will be discussed in detail in the upcoming sections. The SAD has a modular control system, which includes one controller per game mode Moreover, a *standby controller* is required in practice for the SAD in order to support all the active and passive game modes. In the next section, the standby controller will be explained.

Remark 2 In order to control the SAD, in our case studies (see Section 4.1) the interface PyParrot[4] for Python has been used. With this library, in order to control the speed of the drone (i.e., both the displacement and the time the drone needs in order to implement the displacement), five variables can be controlled: duration of the displacement (in seconds), vertical speed of the drone (in m/s), roll, pitch, and yaw displacements of the drone (in deg), all given as a percentage ranging in $[-100, 100]$ of their maximum allowed values, with the signs determining the direction of the corresponding displacement. See Figure 12 for the definition of the roll, pitch, and yaw angles, noting that the axes have been defined such that they are consistent with the definition of the x axis and y axis given earlier on for the SAD's frame of view. In the next sections, the controllers that steer the motions of the SAD in the x and z directions (see Figure 12), generate, respectively, the roll angle and pitch

[4] https://pyparrot.readthedocs.io/

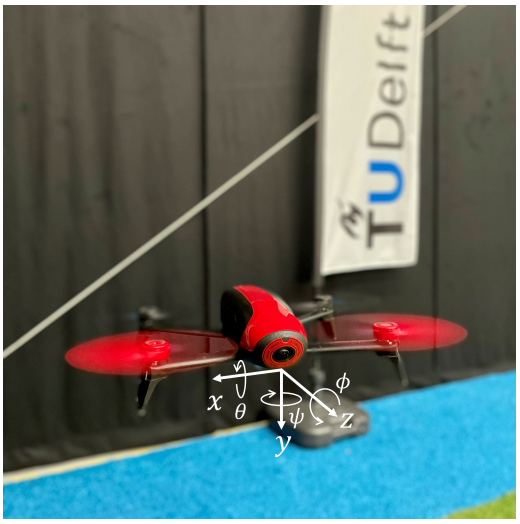


Fig. 12 Reference frame of the SAD with the x-axis pointing to the right, y-axis pointing downwards, and z-axis pointing forwards, and the pitch θ, yaw ψ, and roll ϕ angles being defined accordingly

angle as a percentage of their maximum allowed values. Moreover, Appendix A represents the dynamics of the quadcopter, which formulates the relationships between the lateral and angular displacements of the SAD.

3.4.1 Standby Controller

The standby controller acts as a support module in initializing and sustaining the DMT game modes. The standby controller meets four main objectives:

1. Finding the user in the environment, when the IPU has not been able to locate a user yet
2. Positioning the user in the center of the images captured by the SAD (i.e., in the center of the SAD's frame of view)
3. Maintaining an appropriate, safe distance between the SAD and the user according to a proper distance estimator algorithm
4. Sustaining the altitude of the SAD, such that it fits the corresponding game mode

(1) Finding the user: In case neither of the two ANNs detects a user, the standby controller steers the SAD to rotate with a high yaw rate (e.g., 50% of the maximum allowed yaw displacement within a given time interval, which the SAD takes to execute the corresponding rotation) towards the position that the user has lastly been identified during the current session. In case there is no information about the recent position of the user, the SAD will by default turn to either right or left and will analyze the images captured via the IPU to detect the user. In case the existence of the user cannot be identified via this data, the SAD executes another turn and continues this procedure until the user is detected.

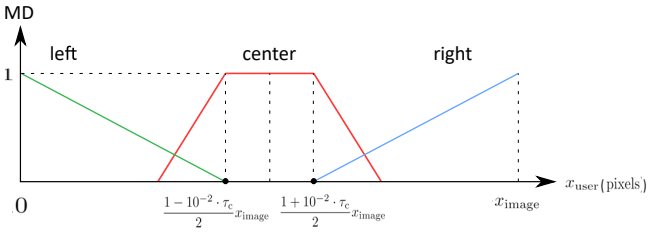


Fig. 13 Membership functions corresponding to fuzzy sets "left", "center", and "right" used by the standby controller to position the user in the frame of view of the SAD

(2) Centering the user in the SAD's frame of view: After a user's position is identified - either as the original input to the standby controller or after the standby controller assists the SAD to locate the user - the standby controller should position the user in the center of the SAD's frame of view according to a pre-specified threshold τ_c. The SAD considers the user to be in the center of its frame of view, whenever the following condition is satisfied:

$$\frac{1-10^{-2}\cdot\tau_c}{2}x_{\text{image}} \leq x_{\text{user}} \leq \frac{1+10^{-2}\cdot\tau_c}{2}x_{\text{image}} \tag{12}$$

where x_{image} is the maximum number of pixels in the horizontal direction on the captured images, and x_{user} is the user's detected position in pixels, considering the user's chest position given by ANN 2, and in case this coordinate is not available, considering the centroid of the user's face-box provided by ANN 1. More specifically, the standby controller identifies the relative position of the user within the frame of view of the SAD using fuzzy sets "left", "center", and "right" with their corresponding membership functions. One example for such membership functions that satisfies (12) is illustrated in Figure 13. The parameter τ_c is generally personalized per user.

The output $v_{\text{SAD}}^{\text{standby}}$ of the standby controller in this case is a percentage of the maximum yaw displacement of the SAD within a given fixed time, which the SAD uses to execute the rotation, and is determined by a TSK-based FL controller approach, which includes the following rules:

$$\begin{aligned}
&\textbf{R1:}\ \text{If}\ x_{\text{user}}\ \text{is}\ \textbf{left},\ \ \text{then}\ v_{\text{SAD}}^{\text{standby}} = a_1 x_{\text{user}} - b_1\\
&\textbf{R2:}\ \text{If}\ x_{\text{user}}\ \text{is}\ \textbf{right},\ \text{then}\ v_{\text{SAD}}^{\text{standby}} = a_2 x_{\text{user}} - b_2\\
&\textbf{R3:}\ \text{If}\ x_{\text{user}}\ \text{is}\ \textbf{center},\ \text{then}\ v_{\text{SAD}}^{\text{standby}} = a_3 x_{\text{user}} - b_3
\end{aligned} \tag{13}$$

where the terms "left", "right", and "center" (more accurately referring to the relative horizontal position of the chest point or face-box centroid of the user w.r.t. the center of the SAD's frame of view) will mathematically be represented by fuzzy sets and their corresponding fuzzy membership function (see, e.g., Figure 13), and parameters a_1, b_1, a_2, b_2, a_3, b_3 will be identified per user according to (12). Figure 14 illustrates the SAD interacting with a user before and

after the standby controller positions the user in the center of the SAD's frame of view.

(3) Maintaining appropriate distance with the user: Sustaining effective interactions with the user for long periods of time is essential for the SAD to achieve satisfactory therapeutic outcomes via DMT. In order to achieve this goal, an adequate distance to the user must be maintained at all times. On the one hand, the SAD should avoid getting too close to the user, since a too small distance raises safety concerns and might intimidate the user, thus negatively impacting the therapeutic interactions. On the other hand, if the distance of the SAD to the user becomes too large, there is a risk that the attention of the user to the SAD is lost and hence, the engagement of the user in the therapeutic interactions decreases. Consequently, the control system of the SAD should constantly receive the detected relative distance of the SAD and the user and adjust it whenever needed.

In case the SAD is not equipped with any sensor that directly measures the longitudinal or lateral distances of the drone to a target (in this case the user), the distance to the user must be estimated by the SAD based on the values gathered and analyzed by the IPU and using a distance estimator algorithm. In order to check whether the relative distance is too small, the values provided by ANN 1 are used. More specifically, based on the width w and height h of the latest face-box (see Figure 3), the area A_{facebox} of the face-box in squared pixels is computed. This area is then divided by the total image area, A_{image}, in squared pixels to determine the face-box ratio.

For two reasons ANN 1 has been used to specify whether the drone is (too) close to the user: (1) the face-box can properly be detected only if the user and the drone are close enough, while for a too close distance ANN 2 may fail to detect the entire body of the user; (2) unlike ANN 2, which requires the detection of the chest and the waistline to make reliable estimates, ANN 1 works as soon as the face of the user is present in the SAD's frame of view.

Whenever the magnitude of the face-box ratio exceeds a certain threshold, $\tau_{\text{d},1}$, the distance of the SAD to the user is considered to be too small. We have:

$$\text{If } \frac{A_{\text{facebox}}}{A_{\text{image}}} > \tau_{\text{d},1}, \text{ SAD is } \textbf{close} \text{ to the user} \qquad (14)$$

where the value of $\tau_{\text{d},1}$ can be identified corresponding to the relative distance between the SAD and the user that feels safe for this particular user. More specifically, the standby controller considers a fuzzy set and its corresponding membership function for the concept of "close" (see, e.g., Figure 15, which shows a membership function for the term "far" defined according to (14)). Generally speaking, the parameter $\tau_{\text{d},1}$ is personalized per user.

To assess whether the SAD is too far from the user, the values provided by ANN 2 are used, since in larger distances the estimates provided by ANN 1 may become unreliable. Out of the 18 joints illustrated in Figure 4, the coordinates the joints 1, 8, and 11 corresponding to the chest and the right and left edges of the user's waistline are considered. Note that joint 1 is the reference joint, i.e., all other joints are detected with respect to joint 1. Therefore, whenever the SAD has information available from ANN 2, the data regarding joint 1 is certainly available. Moreover, while the legs and arms of the human (i.e., the corresponding joints 2-13) are possible to move out of the frame of view of the drone, there is a high chance that the data regarding joints 8 and 11 is available whenever the user is not too close to the SAD. Finally, the joints corresponding to the user's face (i.e., joints 0 and 14-17) are more prone to being swayed and are hence less reliable, especially, when the user moves their neck too frequently.

Once the three coordinates for joints 1, 8, and 11 are known, the vertical distance $\delta_{1,11}$ between joints 1 and 11, and the vertical distance $\delta_{1,8}$ between joints 1 and 8 (see Figure 16) are computed in pixels and are divided by the overall vertical length $h_{\max}$ of the image. We call the resulting values are called the upper body ratios. The closer the user to the SAD, the larger the values of the upper body ratios. More specifically, whenever the maximum of the two values estimated for the upper body ratio is less than a certain threshold $\tau_{\text{d},2}$, then the distance of the SAD to the user is considered too large. We have:

$$\text{If } \max\left\{\frac{\delta_{1,8}}{h_{\max}}, \frac{\delta_{1,11}}{h_{\max}}\right\} < \tau_{\text{d},2}, \quad \text{SAD is } \textbf{far} \text{ from the user} \qquad (15)$$

where the threshold $\tau_{\text{d},2}$ is identified based on the relative distance between the SAD and the user, for which the IPU can still perform satisfactorily, while the user remains attentive to the SAD and hence involved in the DMT interactions ($\tau_{\text{d},2}$ can thus be identified as a personalization parameter in the human-SAD interactions). More specifically, the standby controller considers a fuzzy set and its corresponding membership function for the concept of "far" per user. Figure 17 shows one example with the membership function being defined according to (15).

Remark 3 For the upper body ratio of the user, both $\delta_{1,8}$ and $\delta_{1,11}$ are considered in order to increase the reliability of the estimations. Suppose that one of the two waistline joints is not correctly identified by ANN 2. Since both estimations of the upper body ratio are available, by considering the maximum of the two - see (15) - a more robust evaluation of the relative distance of the SAD and the user can be provided.

Remark 4 The measurements corresponding to the chest position are significantly more reliable than those corresponding to the waistline corners, because the chest position is the reference point of ANN 2.

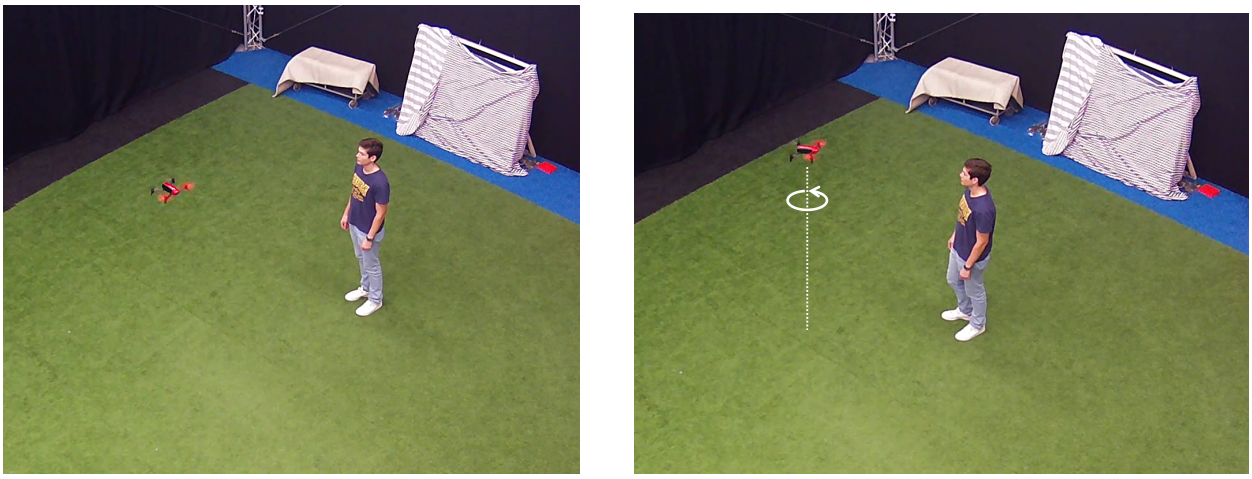


Fig. 14 Performance of the standby controller in centering the user in the SAD's frame of view: In the left-hand side picture, the user is not yet centered. In the right-hand side picture, the standby controller steers the SAD to follow a TSK-based FL controller in order to center the user according to condition (12)

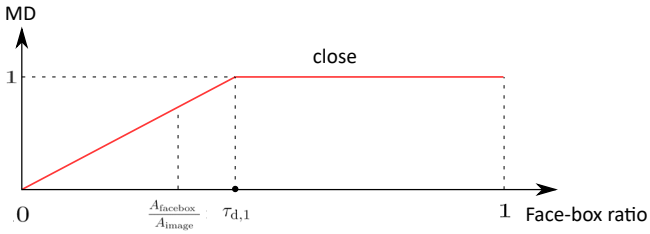


Fig. 15 Membership function corresponding to fuzzy set "close" used by the standby controller to estimate the distance of the user with respect to the SAD

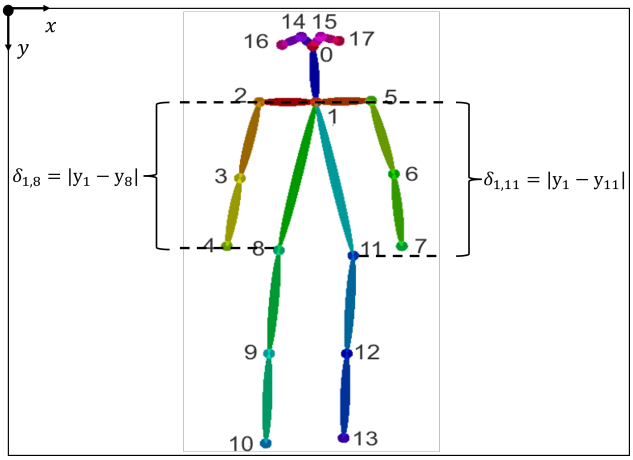


Fig. 16 Illustration of the vertical distances $\delta_{1,8}$ and $\delta_{1,11}$, which are used by the SAD to determine whether its relative distance to the user has become too large

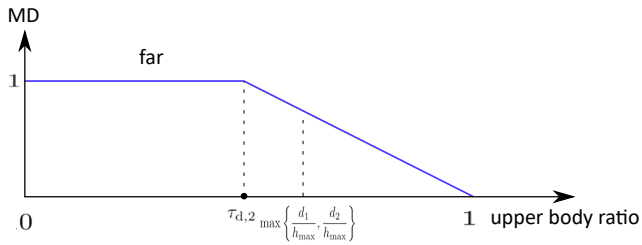


Fig. 17 Membership function corresponding to fuzzy set "far" used by the standby controller to estimate the distance of the user with respect to the SAD

In order to maintain an appropriate and safe distance for interactions with the user, the distance estimator algorithm described above provides an input for the standby controller: "close" (i.e., (14) holds), "far" (i.e., (15) holds), "appropriate and safe" (when neither (14) nor (15) holds). Then a TSK-based FL controller approach according to the following rule base is used to maintain the appropriate or safe distance by determining the SAD's pitch displacement as a percentage $\pi_{\text{SAD}}^{\text{standby}}$ of its maximum allowed value:

R4: If distance is **close**, then $\pi_{\text{SAD}}^{\text{standby}} = a_4 A + b_4$
R5: If distance is **far**,
then $\pi_{\text{SAD}}^{\text{standby}} = a_5 \max\{\delta_{1,8}, \delta_{1,11}\} + b_5$ (16)

where the terms "close" and "far" will mathematically be represented by fuzzy sets and their corresponding fuzzy membership functions and parameters a_4, b_4, a_5, b_5 will be identified per user and according to (14) and (15).

(4) Sustaining the SAD's altitude: Finally, in order to regularly sustain the altitude of the SAD, two thresholds τ_{h} and τ_{l} are defined in order to detect whether the SAD's altitude is *too high* or *too low*. When the SAD maintains an altitude that is too high or too low, two issues may occur. First, the user may not accurately capture all the movements of the SAD and the corresponding coordinates, which negatively impacts the interactions with and the the engagement of the user. Second, the IPU may fail to capture the movements and the corresponding coordinates of the user correctly, which negatively impacts the analysis and decision making of the SAD likewise many quadcopters. Parrot Bebop 2, the drone that is used as the SAD in this research, is equipped with an altitude measurement sensor that will be used in two cases during the DMT sessions: (1) whenever the SAD accomplishes a game mode and transitions to a new game

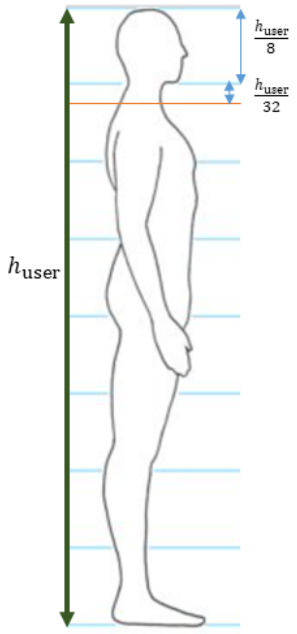


Fig. 18 Ratios of the upper part of the body, i.e., above the shoulders (almost 0.15 of the user's height), which is used to adjust the default altitude of the SAD

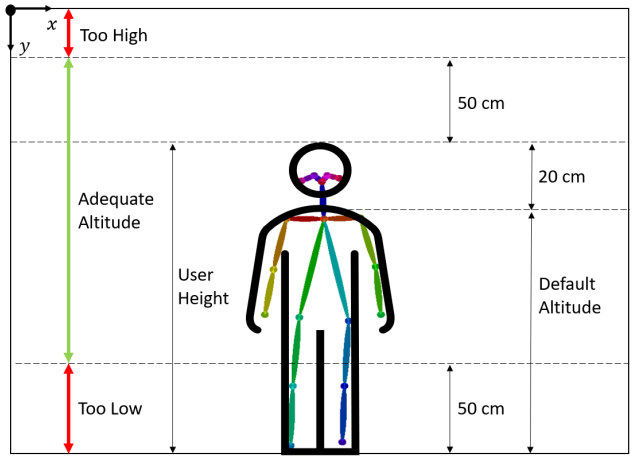


Fig. 19 Illustration of the default altitude and the "too low" and "too high" altitude thresholds for the SAD interacting with a user of height 133.5 cm

mode; (2) just after the SAD is done executing a vertical movement.

To quantify the concepts of "too high" and "too low" for every user, the user's height h_{user} should be taken into account. Generally speaking, we consider the thresholds $\tau_{\text{h}} = \alpha_{\text{h}} h_{\text{user}}$ and $\tau_{\text{l}} = \alpha_{\text{l}} h_{\text{user}}$ in cm, with $\alpha_{\text{h}} > 1$ and $0 < \alpha_{\text{l}} < 1$ being personalization parameters that are identified per user. At the beginning of every game mode or after executing a vertical movement, for an optimal performance the standby controller considers a default altitude, $h^{\text{def}}_{\text{SAD}}$, for the SAD, where this default value corresponds to the user's shoulder level (see Figure 18). For instance, for an average height of 133.5 cm for a 9 year old child, considering the standard ratio of 1.2 to 8 for the upper

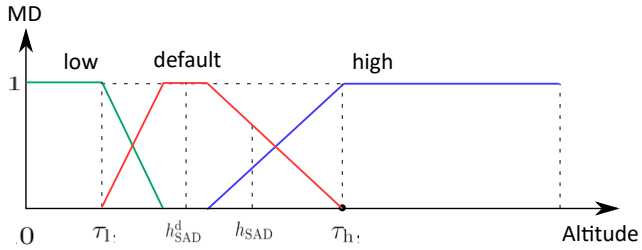


Fig. 20 Membership functions corresponding to fuzzy sets "low", "default", and "high" used by the standby controller to assess the altitude of the SAD

part of the body (above the shoulders) to the entire body length, the SAD should maintain the default altitude of $h^{\text{def}}_{\text{SAD}} = h_{\text{user}} - 20$, with all the magnitudes in cm (see Figure 19).

To maintain the SAD's altitude, the standby controller should first determine the status of the SAD's altitude h_{SAD} according to the following fuzzy rules:

$$\text{If } h_{\text{SAD}} > \tau_{\text{h}}, \text{ SAD's altitude is too } \textbf{high} \tag{17}$$

$$\text{If } h_{\text{SAD}} < \tau_{\text{l}}, \text{ SAD's altitude is too } \textbf{low} \tag{18}$$

$$\text{If } h_{\text{SAD}} \approx h^{\text{def}}_{\text{SAD}}, \text{ SAD's altitude is } \textbf{default} \tag{19}$$

with the terms "high", "default", and "low" mathematically being represented by fuzzy sets and their corresponding fuzzy membership functions (for instance, see Figure 20), where, in general, τ_{h}, τ_{l} and $\approx h^{\text{def}}_{\text{SAD}}$ (to be read as "approximately the default altitude") are identified per user. A TSK-based FL controller is then used according to the following rule base to maintain the proper altitude for the SAD by adjusting its vertical displacement as a percentage $\zeta^{\text{standby}}_{\text{SAD}}$ of its maximum allowed value:

$$\begin{aligned}
&\textbf{R6: If altitude is } \textbf{low}, \text{ then } \zeta^{\text{standby}}_{\text{SAD}} = a_6 h_{\text{SAD}} + b_6 \\
&\textbf{R7: If altitude is } \textbf{default}, \text{ then } \zeta^{\text{standby}}_{\text{SAD}} = a_7 h_{\text{SAD}} + b_7 \\
&\textbf{R8: If altitude is } \textbf{high}, \text{ then } \zeta^{\text{standby}}_{\text{SAD}} = a_8 h_{\text{SAD}} + b_8
\end{aligned} \tag{20}$$

where the parameters a_6, b_6, a_7, b_7, a_8, b_8 will be identified per user.

3.4.2 Passive Game Mode 1: Mirror the User

In passive game modes the SAD mainly responds - via mirroring - to the user's movements. The main goal of passive game modes for the SAD is to accurately replicate all the relevant motions of the user that have been captured during the wait time. These movements include vertical motions, horizontal motions, and their combinations resulting in generally diagonal motions. The motions captured by the IPU are analyzed using the motion processing algorithm (see Section 3.3.1). The SAD then mirrors the motions of the user's chest position. In game mode 1, in order to start the interaction the SAD may emit a speech signal via the

speakers of the remote computer to invite the user to move (e.g., with a piece of music).

Every identified motion category is characterized by three quantities: (1) the time interval (in seconds) associated to the movement, (2) the average vertical speed (in pixels per second) of the movement, and (3) the average horizontal speed (in pixels per second) of the movement. These quantities are the inputs to the corresponding controller for game mode 1, which should steer the SAD to mimic the same motion categories. Moreover, three quantities are output from the controller: (1) the mimicking time, (2) the vertical displacement of the SAD, and (3) the roll displacement of the SAD, such that it results in a desired horizontal displacement.

Remark 5 Diagonal motions may be identified and mimicked by the SAD according to the conditions that are represented in item 2 of Section 3.3.1. Otherwise, the average speed corresponding to the direction (either horizontal or vertical) that is absent in the particular movement is set to zero.

The mimicking time of the SAD is set equal to the time interval associated to the identified motion category. The vertical and horizontal displacements of the drone are considered to be a scaled version of the movement captured from the user. Note that since the data captured by the IPU is in pixels. The corresponding TSK-based FL controller generates the SAD's displacements in cm according to the data registered in pixels.

For the horizontal movements of the SAD in game mode 1, the following TSK-based FL controller determines the percentage $\rho_{\text{SAD}}^{\text{GM1}}$ of the maximum roll displacement (in deg) of the SAD, which should be executed by the SAD for a given fixed time:

$$\begin{aligned}
&\textbf{R9:} \quad \text{If } \dot{x}_{\text{user}}^{\text{chest}} \text{ is } \textbf{small}, \\
&\qquad \text{then } \rho_{\text{SAD}}^{\text{GM1}} = a_9 \dot{x}_{\text{user}}^{\text{chest}} + b_9 \\
&\textbf{R10:} \quad \text{If } \dot{x}_{\text{user}}^{\text{chest}} \text{ is } \textbf{medium}, \\
&\qquad \text{then } \rho_{\text{SAD}}^{\text{GM1}} = a_{10} \dot{x}_{\text{user}}^{\text{chest}} + b_{10} \\
&\textbf{R11:} \quad \text{If } \dot{x}_{\text{user}}^{\text{chest}} \text{ is } \textbf{large}, \\
&\qquad \text{then } \rho_{\text{SAD}}^{\text{GM1}} = a_{11} \dot{x}_{\text{user}}^{\text{chest}} + b_{11}
\end{aligned} \tag{21}$$

with $\dot{x}_{\text{user}}^{\text{chest}}$ the average horizontal speed of the user's chest in pixels/s. The terms "small", "medium", and "large" will mathematically be represented by fuzzy sets and their corresponding fuzzy membership functions, and the parameters a_9, b_9, a_{10}, b_{10}, a_{11}, and b_{11} will be identified per user.

Generally speaking, for the vertical displacement of the SAD one may use a rule base formulated similarly as (21). However, in practice the range of vertical movements of the body compared to the range of horizontal movements is small. Therefore, the rule base (21) in this case reduces to a proportional control relationship that determines a percentage $\zeta_{\text{SAD}}^{\text{GM1}}$ of the maximum allowed vertical speed (in m/s) of the SAD:

$$\zeta_{\text{SAD}}^{\text{GM1}} = K^{\text{GM1}} \dot{y}_{\text{user}}^{\text{chest}} \tag{22}$$

with $\dot{y}_{\text{user}}^{\text{chest}}$ representing the average vertical speed of the user's chest in pixels/s and K^{GM1} a tuning parameter.

Once all the motion categories identified by the SAD have been mimicked, the SAD centers the user within its frame of view using the standby controller (see Section 3.4.1) and waits to capture any new motions. In a personalized DMT plan, the number of sequences the SAD allocates to game mode 1 is user-specific, which is identified by the SAD in the course of long-term interactions with the user and according to the assessment of the performance and engagement of the user in the corresponding game mode. For instance, if a user does not engage in the interactions with the SAD for a specific number of consecutive sequences, the SAD initiates a new game mode. Figures 21 and 22 illustrate two motion mimicking sequences, where the SAD follows the user who moves to the, respectively, right and left from the SAD's perspective.

3.4.3 Passive Game Mode 2: Mirror the User's Hand

In game mode 2 the SAD mirrors the movements associated to the user's hands. Similarly to game mode 1, the standby controller initially positions the user in the centre of the SAD's frame of view, maintains a safe and engaging distance from the user, and sustains the SAD's altitude. The SAD encourages the user by emitting the speech "waiting for your hand motion!" and waits for the user to initiate the game by moving their hands. Then during the wait time, which is user-specific, the SAD captures and analyzes all the movements of the user's hands. There are two main differences in game mode 2 compared to game mode 1:

1. The positions r_{t_i} that are recorded and analyzed by the IPU correspond to the wrist positions provided by ANN2 (see joints 4 and 7 in Figure 4). The controller corresponding to game mode 2 first decides which hand to follow by estimating the total distance (in pixels) that is travelled by each wrist during the wait time. The hand that has been more active (i.e., the total travelled path in pixels of the corresponding wrist within the wait time has been larger) is selected.
2. Compared to the chest point, capturing the wrist positions is more prone to failures or errors for the IPU. More specifically, the pace of the hand movements may be according to frequencies that are higher than the frequency of capturing data by the IPU in real-time applications. Such hand movements, therefore, are not captured and perceived correctly by the IPU and result in missing or erroneous data. Therefore, the fault tolerance algorithm explained in Section 3.3.1, may more frequently be called by the controller in game mode 2.

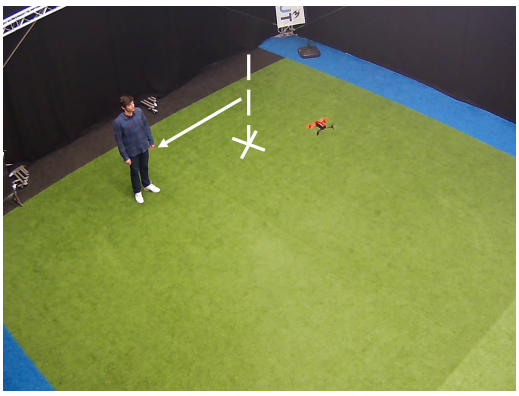
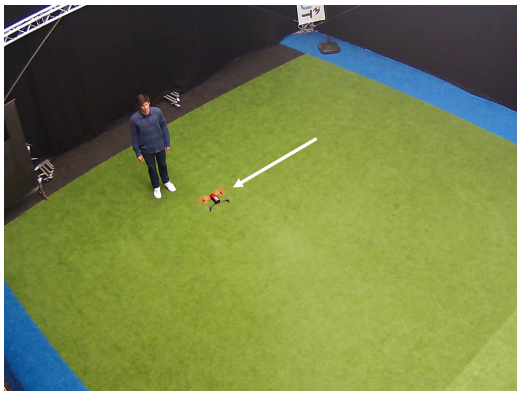



Fig. 21 Passive game mode 1: The user is originally standing in front of the SAD (the standby controller has centered the user's face-box in the SAD's frame of view); the user moves to the **left** from the perspective of the SAR (see the left-hand side picture), and the SAD mimics the same motion (see the right-hand side picture)

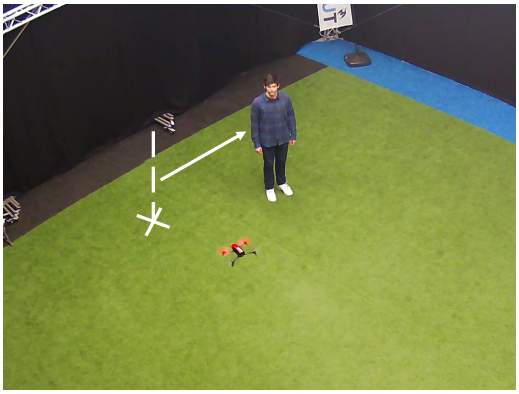
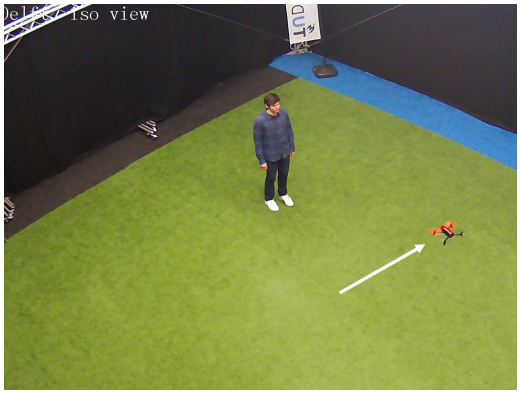



Fig. 22 Passive game mode 1: The user is originally standing in front of the SAD (the standby controller has centered the user's face-box in the SAD's frame of view); the user moves to the **right** from the perspective of the SAR (see the left-hand side picture), and the SAD mimics the same motion (see the right-hand side picture)

The TSK-based FL controller corresponding to game mode 2 performs according to the following rule base for the horizontal movements of the SAD:

$$\begin{aligned}
&\textbf{R12:}\ \text{If } \dot{x}_{\text{user}}^{\text{wrist}} \text{ is } \textbf{small}, \\
&\quad\ \text{then } \rho_{\text{SAD}}^{\text{GM2}} = a_{12}\dot{x}_{\text{user}}^{\text{wrist}} + b_{12} \\
&\textbf{R13:}\ \text{If } \dot{x}_{\text{user}}^{\text{wrist}} \text{ is } \textbf{medium}, \\
&\quad\ \text{then } \rho_{\text{SAD}}^{\text{GM2}} = a_{13}\dot{x}_{\text{user}}^{\text{wrist}} + b_{13} \\
&\textbf{R14:}\ \text{If } \dot{x}_{\text{user}}^{\text{wrist}} \text{ is } \textbf{large}, \\
&\quad\ \text{then } \rho_{\text{SAD}}^{\text{GM2}} = a_{14}\dot{x}_{\text{user}}^{\text{wrist}} + b_{14}
\end{aligned} \tag{23}$$

with $\dot{x}_{\text{user}}^{\text{wrist}}$ the average horizontal speed of the user's (more active) wrist in pixels/s, and $\rho_{\text{SAD}}^{\text{GM2}}$ the percentage of the maximum roll displacement (in deg) of the SAD computed by the controller for game mode 2. The terms "small", "medium", and "large" will mathematically be represented by fuzzy sets and their corresponding fuzzy membership functions, and the parameters a_{12}, b_{12}, a_{13}, b_{13}, a_{14}, and b_{14} will be identified per user.

For the vertical displacements of the SAD, a proportional control relationship provides a percentage $\zeta_{\text{SAD}}^{\text{GM2}}$ of the maximum allowed vertical speed (in m/s) of the SAD to steer it according to passive game mode 2:

$$\zeta_{\text{SAD}}^{\text{GM2}} = K^{\text{GM2}}\dot{y}_{\text{user}}^{\text{wrist}} \tag{24}$$

where $\dot{y}_{\text{user}}^{\text{wrist}}$ is the average vertical speed of the user's wrist in pixels/s for the active hand.

Figures 23-26 illustrate experiments performed in the Cyber Zoo, where the controller corresponding to game mode 2 steers the SAD to mimic the motions of the user's right hand.

Remark 6 In game mode 1, the parameters of the corresponding TSK-based FL controller can be tuned such that the SAD mimics the movements of the user's body accurately (i.e., with scale 1). In game mode 2, however, the parameters corresponding to the TSK-based FL controller may be tuned such that the resulting movements of the SAD are scaled (with a factor usually larger than 1) compared to the movements of the user's hand. This is mainly to stress the movements of the user in game mode 2 (for a more engaging interaction).

3.4.4 Active Game Mode 3: Mimic the SAD with Body Motions

Game modes 3 and 4 are active, meaning that the SAD interacts with the user by initiating movements that the user should mimic. After centering the user

Fig. 23 Passive game mode 2: The user initiates a horizontal movement with the right hand

Fig. 24 Passive game mode Mode 2: The SAD mimics (a scaled version of) the user's hand motion

in the frame of view of the SAD and maintaining an adequate distance between the SAD and the user using the standby controller, the SAD emits a sound, e.g., "*Follow me with your body*" (for game mode 3) or "*Follow me with your hand*" (for game mode 4), via the speakers of the remote computer to inform the user about initiating a movement.

In active game mode 3, the SAD's motions consist of pure horizontal motions (i.e., ample sliding to the left or right), pure vertical motions, or a simultaneous combination of the two, which results in diagonal motions. The range of these movements should be personalized per user. The direction of the motions is initially selected in a random way. After executing a movement, the SAD pauses according to the *wait time* to give the user the chance to follow the SAD's movement. Afterwards, using the motion processing algorithm, the SAD analyzes and assesses the user's data that is captured via the IPU, and attributes a *performance score* to the analyzed movement of the user, which implies how well the motion of the SAD has been mimicked by the user. In order to further stimulate the interactions with the user, the TSK-based FL controller corresponding to game mode 3 responds to higher performance scores with a larger displacement magnitude. Gradually, the frequency, category, duration, direction, and speed of the SAD's movements are

Fig. 25 Passive game mode 2: The user initiates a diagonal motion with the left hand

Fig. 26 Passive game mode 2: The SAD mimics (a scaled version of) the diagonal motion of the user's hand

adapted according to the responses received from the user and the performance scores attributed.

The input to the TSK-based FL controller of game mode 3 is the horizontal distance Δx_{user} in pixels between the last registered position x_{user} of the user and the center of the SAD's frame of view. The output $\rho_{\text{SAD}}^{\text{GM3}}$ of the TSK-based FL controller of game mode 3 is a percentage of the maximum allowed roll displacement of the SAD. This controller consists of the following rule base for the horizontal movements of the SAD:

$$\begin{aligned} &\textbf{R15: If } \Delta x_{\text{user}} \text{ is } \textbf{negligible}, \\ &\qquad \text{then } \rho_{\text{SAD}}^{\text{GM3}} = a_{15}\Delta x_{\text{user}} + b_{15} \\ &\textbf{R16: If } \Delta x_{\text{user}} \text{ is } \textbf{significant}, \\ &\qquad \text{then } \rho_{\text{SAD}}^{\text{GM3}} = a_{16}\Delta x_{\text{user}} + b_{16} \end{aligned} \tag{25}$$

where the terms "negligible" and "significant" should mathematically be represented by fuzzy membership functions and parameters a_{15}, b_{15}, a_{16}, and b_{16} should be tuned according to the preferences and responses of every user.

A proportional control policy, similar to the previous game modes, may also be considered to steer the vertical movements of the SAD. Figures 27-29 illustrate a sequence of movements corresponding to game mode 3 for the SAD and a user in the Cyber Zoo. In Figures 27 the SAD has centered the user's image in its frame of view and has maintained a proper distance with the user via the standby controller. In Figure 28, the SAD moves to the left-hand side of the user, and waits for the user to mimic this motion. In Figure 29 the user moves in the same direction while the SAD captures the user's motion and evaluates it to give it a performance score.

3.4.5 Active Game Mode 4: Mimic the SAD with Hand Motions

In game mode 4 the SAD expects the user to follow its movements by hand. The TSK-based FL controller corresponding to game mode 4 for generating the horizontal movements of the SAD consists of the following rule base:

$$\begin{aligned} &\textbf{R17: If } \Delta x_{\text{hand}} \text{ is } \textbf{negligible}, \\ &\quad \text{then } \rho_{\text{SAD}}^{\text{GM4}} = a_{17}\Delta x_{\text{hand}} + b_{17} \\ &\textbf{R18: If } \Delta x_{\text{hand}} \text{ is } \textbf{significant}, \\ &\quad \text{then } \rho_{\text{SAD}}^{\text{GM4}} = a_{18}\Delta x_{\text{hand}} + b_{18} \end{aligned} \tag{26}$$

where the terms "negligible" and "significant" will be represented by fuzzy membership functions and parameters a_{17}, b_{17}, a_{18}, and b_{18} will be tuned. Note that Δx_{hand} is the horizontal distance between the wrist of the user and the center of the SAD's frame of view. Whenever data regarding both hands of the user is available, the hand that is closer to the center of the SAD's frame of view will be considered.

For the vertical movements of the SAD, a proportional controller given below is used:

$$\zeta_{\text{SAD}}^{\text{GM4}} = \min\left\{K^{\text{GM4}} \cdot \frac{1}{\Delta y_{\text{hand}}}, \zeta_{\text{SAD}}^{\max}\right\} \tag{27}$$

where $\zeta_{\text{SAD}}^{\text{GM4}}$ is a percentage of the maximum vertical speed (in m/s) of the SAD, $\zeta_{\text{SAD}}^{\max}$ further limits the maximum vertical displacements of the SAD during game mode 4 (note that since hands are expected to mirror more abrupt sequences of motions of the SAD, the scale of these motions - in both horizontal and vertical directions - have been scaled down), and Δy_{hand} is the vertical distance between the user's hand and the center of the drone's frame of view. Also note that a larger value for Δy_{hand} corresponds to a worse performance for the user, and in response to that the SAD makes a smaller displacement. Therefore, 27 provides an inverse relationship between Δy_{hand} and the corresponding output of the controller.

Fig. 27 Active game mode 3: The SAD uses the standby controller to center the user's image in its frame of view and to maintain a proper distance with the user, and emits a sound "follow me with your body" to initiate the active game mode

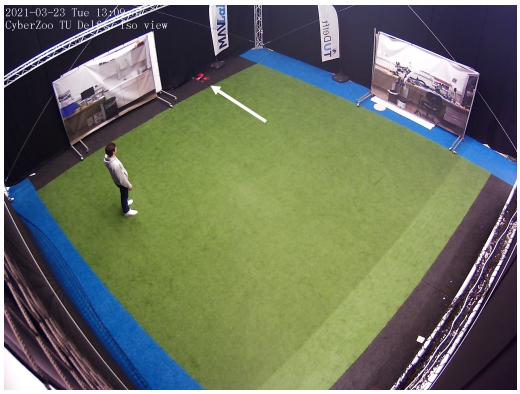


Fig. 28 Active game mode 3: The SAD moves to the left-hand side of the user and pauses

Fig. 29 Active game mode 3: While the user is mimicking the SAD, the SAD captures the user's movements via its IPU and assesses the user's performance

Figures 30-32 illustrate a sequence of interactions between the SAD and a user according to active game mode 4, where the user mimics the SAD's movements by their hand.

3.5 Adaptivity and Personalization of the SAD

As discussed before, adaptivity and personalization are important characteristics that are essential for the SAD's control system. In this section we explain in detail how adaptivity and personalization are incorpo-

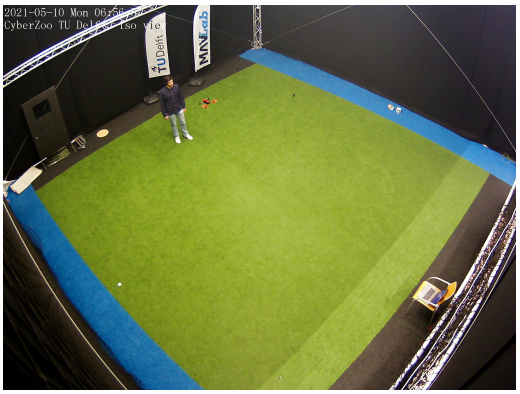


Fig. 30 Active game mode 4: The SAD uses the standby controller to center the user's image in its frame of view and to maintain a proper distance with the user, and emits a sound "follow me with your hand" to initiate the active game mode

Fig. 31 Active game mode 4: The SAD moves to the left-hand side of the user and waits for the user's left hand to mirror this motion

Fig. 32 Active game mode 4: The SAD moves to the right-hand side of the user and waits for the user's right hand to mirror this motion

rated into the SAD's controllers of passive and active game modes based on metrics such as the user's performance and engagement level.

Adaptivity refers to the possibility of updating the parameters that identify a DMT plan (e.g., the wait time of the SAD) or parameters of the controllers of the SAD according to the general conditions of users and the DMT sessions. Personalization refers to the procedure of tuning such parameters or selecting the order, frequency, and duration of various game modes, in a user-specific way. While parameters that are personalized via interactions with a user may remain constant for that user after being identified, parameters that are adaptive may vary more frequently. It is mainly a decision of the designer to specify a tuning parameter as one of these two categories. Note that some of the adaptive parameters may be a function of personalization parameters (e.g., if the SAD's wait time is considered as an adaptive parameter that may vary in the course of one DMT session, its upper or lower values are still determined based on the preference of every specific user).

We first discuss the adaptivity of the controllers. Different adaptivity modules were developed for passive and active game modes, based on two main metrics, i.e., the performance and engagement of the user. The performance of the user is evaluated based on how well the user has accomplished a particular task (i.e., initiating or mirroring) corresponding to a specific game mode, while engagement is based on whether or not a user is attempting to interact with the SAD. A high performance implies a high level of engagement, while the opposite is not necessarily true.

On the one hand, the adaptivity policies that are developed lead the controller to act more leniently towards a user, who exhibits a low performance but a high engagement. On the other hand, for a high performance (which also implies a high engagement), the developed adaptivity approaches steer the game modes to become more challenging for the user.

3.5.1 Adaptivity for Passive Game modes

For passive game modes a trial is defined as the following sequence: the SAD pausing for the user according to the wait time to move, the SAD processing the captured movements, the SAD mimicking the analyzed movements. Moreover, an episode is the total number of trials in one continuous game mode.

Both the performance and level of engagement in passive game modes are quantified via the number of inactive trials. An inactive trial is one, for which no significant (according to a specific threshold, e.g., a multiple of τ_{m}) motions have been detected during the wait time, which implies a performance and a low engagement. A good performance is associated with low numbers of inactive trials.

Passive game modes involve two adaptive parameters: the wait time t_{w} per trial and the number N_{trial} of trials per episode (i.e., the number of chances the SAD gives to a user in one episode to perform a motion according to a particular game mode), where these adaptive parameters are updated at the end of each episode. The wait time is adjusted according to a metric called the episodic average pause time, T_{p}, which is the average time from when the user finishes the action, which the SAD should mimic, until the end of the current wait time.

In practice, there is a desired time interval $\left[T_{\mathrm{p}}^{\mathrm{l}}, T_{\mathrm{p}}^{\mathrm{u}}\right]$, to which T_{p} should belong. In case $T_p > T_{\mathrm{p}}^{\mathrm{u}}$, the wait time is decreased according to (28), and in case $T_p < T_{\mathrm{p}}^{\mathrm{l}}$ the wait time is increased according to (29). We have:

$$t_{\mathrm{w}} \leftarrow \max\left\{t_{\mathrm{w}} - (T_{\mathrm{p}} - T_{\mathrm{p}}^{\mathrm{u}}), t_{\mathrm{w}}^{\mathrm{l}}\right\} \tag{28}$$

$$t_{\mathrm{w}} \leftarrow \min\left\{t_{\mathrm{w}} + (T_{\mathrm{p}}^{\mathrm{l}} - T_{\mathrm{p}}), t_{\mathrm{w}}^{\mathrm{u}}\right\} \tag{29}$$

where $t_{\mathrm{w}}^{\mathrm{l}}$ and $t_{\mathrm{w}}^{\mathrm{u}}$ are the lower and upper values considered for the wait time in a passive game mode. Note that $T_{\mathrm{p}}^{\mathrm{l}}$, $T_{\mathrm{p}}^{\mathrm{u}}$, $t_{\mathrm{w}}^{\mathrm{l}}$, and $t_{\mathrm{w}}^{\mathrm{u}}$ are personalization parameters (i.e., they are user-specific).

The number N_{trial} of trials per episode for a passive game mode is updated according to the number of inactive trials. In particular, whenever the number of inactive trials is equal to or larger than a threshold N_{in}, then N_{trial} is reduced by 1. Alternatively, whenever the number of inactive trials is smaller than N_{in}, then the user is considered to be fully engaged in the game mode, and N_{trial} is increased by 1. Moreover, to prevent too many repetitions or significant avoidance of a game mode (which may not encourage the user to move outside of their comfort zone), a lower $N_{\mathrm{trial}}^{\mathrm{l}}$ and an upper $N_{\mathrm{trial}}^{\mathrm{u}}$ value for N_{trial} may be considered, which can be fixed for all users or can be user-specific.

3.5.2 Adaptivity for Active Game modes

The main goal of the adaptive module in active game modes is to ensure that, on the one hand, the motions performed by the SAD challenge the user and result in an increased engagement level and, on the other, these motions are always feasible for the user to mimic, and do not increase the risk of crashes that endanger the user or damage the SAD.

For an active game mode, a trial is defined as the following sequence: the SAD initiating a motion, the SAD pausing according to the wait time for the user to follow the motion, the SAD re-centering the user in its frame of view using the standby controller. After each trial, the performance and engagement of the user for that trial are evaluated. The total number of trials in one game mode is called an episode.

At the end of every episode, the adaptive parameters may be updated, based on the average episodic performance $\bar{p}_{\mathrm{ep}}$ of the user (i.e., the mean value of the performance scores corresponding to all the trials in that episode) and the average episodic engagement $\bar{e}_{\mathrm{ep}}$ of the user. Note that in active game modes, a good performance for the user is identified whenever the user's final position during the wait time (i.e., the chest position in active game mode 3 and the wrist position in active game mode 4) is close enough (i.e., is less than a threshold τ_{c} introduced in Section 3.4.1) to the center of the SAD's frame of view. Similarly, the engagement level may be quantified according to the performance score. More specifically, whenever the performance score is considered as high (according to τ_{c}), the engagement level is also high. Otherwise, the engagement level receives a low score, unless - despite a poor performance score - the user exhibits noticeable activity or large movements (i.e., displacements with an amplitude larger than a multiple of τ_{m} introduced in Section 3.3.1).

Remark 7 Whenever the average episodic engagement $\bar{e}_{\mathrm{ep}}$ is larger than a specific value (e.g., 0.5) the adaptivity procedure is triggered. Otherwise, the personalization module (explained in Section 3.6) decides to skip that game mode due to the excessive lack of interest of the user.

For the active game modes, the parameters that specify the membership functions corresponding to the terms "negligible" and "significant" in (25) and (26) are considered as adaptive parameters. For instance, if Gaussian membership functions are used, the standard deviations σ_{neg} and σ_{sig} will be updated according to the following relationships:

$$\sigma_{\mathrm{neg}} \leftarrow \max\left\{\sigma_{\mathrm{neg}} - \Delta\sigma, \sigma^{\mathrm{l}}\right\} \tag{30}$$

$$\sigma_{\mathrm{sig}} \leftarrow \min\left\{\sigma_{\mathrm{sig}} + \Delta\sigma, \sigma^{\mathrm{u}}\right\} \tag{31}$$

with σ^{l} and σ^{u} lower and upper values for the standard deviations that may be personalized per user. In general, when the average episodic performance $\bar{p}_{\mathrm{ep}}$ is high, the adaptive module tends to make the game mode more challenging for the user by providing a more strict definition for the terms "negligible" and "significant", which is realized by decreasing σ_{neg} and increasing σ_{sig}. We define:

$$\Delta\sigma = \alpha_\sigma \bar{p}_{\mathrm{ep}} + \beta_\sigma \tag{32}$$

where α_σ and β_σ are personalized (or are determined as fixed values) based on real-life DMT interactions.

Moreover, the parameters a_{17}, a_{18}, b_{17}, and b_{18} in in (25) and (26) are considered as adaptive parameters for active game modes, and will be updated according to the following relationships:

$$a_i \leftarrow \min\left\{ \max\left\{\lambda_a a_i, a_i^{\mathrm{l}}\right\}, a_i^{\mathrm{u}}\right\} \tag{33}$$

$$b_i \leftarrow \min\left\{ \max\left\{\lambda_b b_i, b_i^{\mathrm{l}}\right\}, b_i^{\mathrm{u}}\right\} \tag{34}$$

with $i = 15, 16, 17, 18$ and:

$$\lambda_a = \alpha_a \bar{p}_{\mathrm{ep}} + \beta_a, \qquad \text{where } |\alpha_a| < 1,\ |\beta_a| < 1 \tag{35}$$

$$\lambda_b = \alpha_b \bar{p}_{\mathrm{ep}} + \beta_b, \qquad \text{where } |\alpha_b| < 1,\ |\beta_b| < 1 \tag{36}$$

Note that the upper and lower values a_i^{u}, b_i^{u}, a_i^{l}, and b_i^{l} are determined such that unsafe movements of the SAD are prevented. Moreover, the personalization parameters α_a, β_a, α_b, and β_b are determined in such a way that the slopes of the output of the corresponding controllers provide larger movement amplitudes in the upcoming trials, whenever the user's average episodic

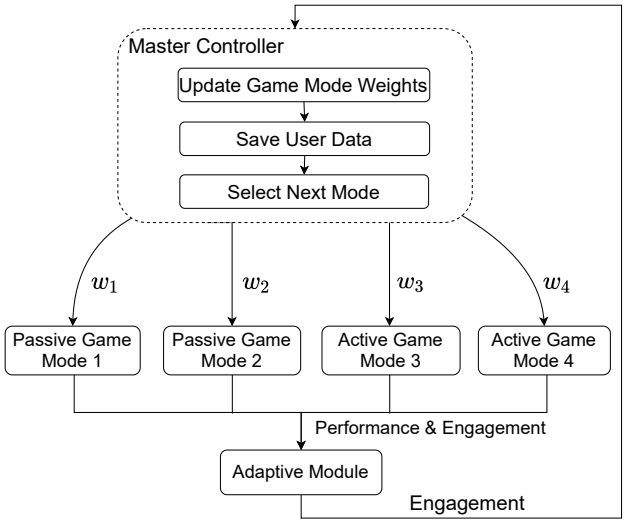


Fig. 33 Controller interaction scheme

performance is high, and otherwise reduce the amplitude of the next movements.

For active game mode 4, where vertical movements are also incorporated into the movements of the SAD, the parameter K^{GM4} in (27) is also considered as an adaptive parameter that will be updated according to the following relationship:

$$K^{\mathrm{GM4}} \leftarrow \min\left\{ \max\left\{\lambda^{\mathrm{GM4}} \cdot K^{\mathrm{GM4}}, K^{\mathrm{l}}\right\}, K^{\mathrm{u}}\right\} \quad (37)$$

with

$$\lambda^{\mathrm{GM4}} = \alpha^{\mathrm{GM4}}\bar{p}_{\mathrm{ep}} + \beta^{\mathrm{GM4}} \quad (38)$$

and with K^{u}, K^{l}, α^{GM4}, and β^{GM4} personalization parameters.

Similarly to passive game modes, in active game modes the wait time t_{w} is considered as an adaptive parameter that will be updated based on the following relationship:

$$t_{\mathrm{w}} \leftarrow \min\left\{ \max\left\{t_{\mathrm{w}} + \alpha_{\mathrm{w}}\bar{p}_{\mathrm{ep}} + \beta_{\mathrm{w}}, t_{\mathrm{w}}^{\mathrm{l}}\right\}, t_{\mathrm{w}}^{\mathrm{u}}\right\} \quad (39)$$

with α_{w}, β_{w}, $t_{\mathrm{w}}^{\mathrm{l}}$, and $t_{\mathrm{w}}^{\mathrm{u}}$ personalization parameters.

Finally, the parameter τ_{c} is considered as an adaptive parameter for active game modes and will be updated according to the following equation:

$$\tau_{\mathrm{c}} \leftarrow \min\left\{ \max\left\{\tau_{\mathrm{c}} - \alpha_{\mathrm{c}}\bar{p}_{\mathrm{ep}} + \beta_{\mathrm{c}}, \tau_{\mathrm{c}}^{\mathrm{l}}\right\}, \tau_{\mathrm{c}}^{\mathrm{u}}\right\} \quad (40)$$

where $\alpha_{\mathrm{c}}, \beta_{\mathrm{c}} > 0$ and $\alpha_{\mathrm{c}} > \beta_{\mathrm{c}}$. Whenever a user exhibits a high average episodic performance, τ_{c} varies such that the control system of the SAD becomes more strict in scoring the user's performance.

3.6 Personalization via a Master Controller

The personalization module of the SAD is represented via a master controller (see Figure 33). Considering episodes consisting of a fixed number of trials, at the end of every episode, the master controller should decide which game mode to be executed (note that a game mode may be selected several times consecutively). Therefore, a non-zero weight w_i is assigned to each game mode i such that $\sum_{i=1}^{4} w_i = 1$. The weight w_i corresponds to the chance of game mode i to be selected by the master controller at the end of an episode.

At the end of every episode, the weights will be updated such that a game mode that has proven to be highly engaging for the user receives a higher weight. The following relationship is used to update the values of the weights for $i = 1, 2, 3, 4$ and $j \in \{1, 2, 3, 4\}/\{i\}$:

$$w_i \leftarrow \min\left\{ \max\left\{w_i + (-1)^{\ell}\Delta w, w^{\min}\right\}, 1\right\} \quad (41)$$

$$w_j \leftarrow \min\left\{ \max\left\{w_j - (-1)^{\ell}\frac{\Delta w}{3}, w^{\min}\right\}, 1\right\} \quad (42)$$

where $\ell = 0$ for $\bar{e}_{\mathrm{ep}}$ above a given threshold, and 1 otherwise. Moreover, Δw is a design parameter and $w^{\min}$ is a minimum value considered for the weight of each game mode, where usually $w^{\min} > 0$ in order to prevent a specific game mode to be completely excluded from the DMT plans.

At the beginning of the next episode, the master controller selects a game mode based on the weight values. In order to avoid repeating the same game mode continuously, after a certain number of episodes, the master controller selects one of the remaining three game modes according to their weight values.

In addition to updating and storing the weights corresponding to various game modes, the master controller also stores and makes use of the personal information of the users that affect the decisions of the SAD, e.g., their height. Moreover, the master controller stores a brief report from every DMT session per user, including their average episodic performance values, the number of inactive trials, the profiles for the weights of various game modes, and the level of engagement of the users for each game mode.

4 Case Study

In this section, we introduce the setup and main facilities that have been used in the experiments for DMT via the proposed SAD and we explain the experiments that have been executed.

Fig. 34 Top view of a Parrot Bebop 2 drone

4.1 Drone Used as a SAD

A Parrot Bebop 2 drone[5] illustrated in Figure 34 was used as the SAD in all the experiments. Parrot Bebop 2 is a small quadcopter, with the dimensions 382 mm (frontal length) × 328 mm (side width) × 89 mm (height), weighing 500 g and equipped with a 2700 mAh battery. This battery power enables the drone to perform DMT for continuous periods of time of up to 10 min. Moreover, two additional fully charged batteries with the same capacity were used to replace the SAD's battery whenever it ran out of charge. This way we ran sessions of up to 30 min per user. Parrot Bebop 2 has a 14 MP camera, which is capable of recording 1080 p video at 30 fps, as well as the drone's own WiFi network, which enables the drone to connect to other devices, e.g., laptops or smartphones. Overall, the trade-off between the size and and the battery power, safe propellers, high quality of the camera, and the simple appealing appearance of the drone makes Parrot Bebop 2 suitable for this research and in general as a SAD.

4.2 Experimental Setup

All the simulated DMT sessions with the SAD and one person at a time as the user have been conducted at the Cyber Zoo[6] of the Delft University of Technology. Cyber Zoo is a research and test laboratory in the Faculty of Aerospace Engineering that embeds a 10 m × 10 m synthetic turf surrounded by safety nets for protecting both participants and robots during the experiments. Furthermore, the experimental facilities at the Cyber Zoo are equipped with twelve high-tech cameras, which in our research were used to record and analyse the SAD's behaviour during the simulated DMT sessions.

The computations corresponding to the online data analysis and online decision making of the SAD are performed off-board on a remote computer that is connected to the SAD via its WiFi network. A main reason, in addition to saving more battery power for a longer performance of the SAD, is that running the computations off-board allows to significantly increase the computational power that is available for real-time image processing and decision making of the SAD. Furthermore, using a remote computer for off-board processing and computations, allows for using a high-level, interpreted programming language such as Python for developing and implementing the SAD's control system. In particular, the Pyfuzzylite library [26] is used in this research. Moreover, using Python simplifies the interface that will be used by, e.g., therapists or caregivers, to control the SAD, as one can use either the official Olympe library [27] of Parrot or alternative libraries that are based on this library, such as Pyparrot library [28] (which was used in this research).

Additionally, the SAD may interact with the user (or with therapists and caregivers) via speech. The sound of the remote computer system in such cases can be used in order to enable the SAD to quickly communicate with humans, e.g., to prompt or to encourage the user to move. Any sound that is emitted in the code is transmitted using either the built-in speaker of the remote computer or via a headset that can be worn by the user. Finally, the SAD should continuously use an Image Processing Unit (IPU) online, which will also run on the remote computer system.

4.3 Parameters

The maximum number of horizontal and vertical pixels in the images captured by the SAD are 856 and 480, respectively. The maximum yaw rate of the SAD is 80 deg/s and its maximum vertical speed is 1 m/s. The time allocated to each movement of the SAD is 1 s and the maximum allowed roll and pitch displacements during this time are 20 deg, which provide safety by preventing the SAD from moving too fast.

In order to assess and score the performance of the user in active game modes 3 and 4, three performance scores, 0, 0.5, and 1 were used. The user receives a score 1 whenever the final position (of the body or the hand) of the user is considered to be centred in the frame of view of the SAD, that is the chest position (or, alternatively, the face-box centroid) of the user is in the proximity of $x = 428$ pixels. Note that the concepts of *close* and hence, *centered* depend on τ_c (see (12)) and are user specific. A score of 0.5 is given to the user whenever the user has moved in the right direction, although the last captured position after the wait time may not be perfectly (with regard to the tolerance τ_c) centered. In case neither of the above cases occurs, the user receives a score of 0.

[5] https://support.parrot.com/pt/en/support/products/parrot-bebop-2

[6] https://tudelftroboticsinstitute.nl/labs/cyber-zoo

4.4 Participants

In total, 10 participants took part in the experiments. For every participant, 3 sets of experiments composed of sessions lasting for 8-12 min were considered. The reason for selecting 3 sets of experiments was to cover more environmental and personal variations corresponding to each participant in order to make sure that both the personalization and adaptation modules were assessed properly. Moreover, in every subsequent experiment, the users became more familiar (and mainly more comfortable) with the setup of the sessions and the SAD itself. The age and height of participants varied between, respectively, 21 years and 24 years and 155 cm and 185 cm. Before the experiments, participants were informed about the nature of the four game modes and about that the SAD's behaviour was adaptive with respect to their preferences. Participants were also informed that they were not obliged to continuously interact with the SAD during the sessions.

Initially all the four game modes were executed during interactions with the participants, so that they get acquainted with all the game modes. Moreover, the SAD executed adaptation and personalization at this stage for each user. Afterwards, the SAD continued the session by autonomously selecting the game modes in accordance to the user's probability profile explained in Section 3.6. In case unwanted errors (e.g., an IPU crash) occurred during an experiment, that experiment was discarded from the results.

4.5 Implementing the IPU

In order to train the ANNs, the COCO dataset[7] has been used. Before implementing the motion processing algorithm and the SAD's controller in real-life experiments, since they receive the output of the IPU as input, the real-time performance of the IPU based on the frequency of image processing (defined as the inverse of the time elapsed to analyze two consecutive images) was evaluated.

Figure 35 illustrates the frequency of the SAD's camera (i.e., the reciprocal of the time required by the camera to capture two consecutive images) for 100 sample images, which shows an average frequency of almost 38 Hz. Moreover, the frequency of capturing the images is always above 20 Hz.

The frequency of the IPU consisting of ANN 1 is illustrated in Figure 36. As a result of incorporating ANN 1 in the IPU, the average frequency of the image processing procedure has significantly dropped to less than one-third (almost 11 Hz).

By incorporating only ANN 2 within the IPU, the average frequency of the image processing (see Figure 37) has dropped to almost 2.5 Hz. The significant discrepancy between the average frequency of the image processing for ANN 1 and ANN 2 is mainly because ANN 2 provides more detailed information regarding the user's position by estimating the accurate position of 18 joints. The importance of this detailed and accurate information for the performance of the SAD and for sustaining a purposeful interaction with the user makes this trade-off between the computation time and accuracy of the IPU's output acceptable.

Finally, by incorporating both ANN 1 and ANN 2 in the IPU, the frequency of the image processing (see Figure 38) becomes almost 2 Hz, i.e., slightly less than that of the IPU with only ANN 2. Moreover, Figure 38 shows that the image processing frequency often takes values between 1.5 Hz and 2 Hz. Therefore, we have decided to develop the motion processing algorithm and the SAD's controller such that they perform properly for frequencies between 1.5 Hz and 2 Hz (i.e., a time interval of 0.5 s to 0.7 s between two IPU estimations).

4.6 Implementing the Motion Processing Algorithm

In this section, we provide further information about the implementation and values used for the parameters that are involved in the motion processing algorithm explained in Section 3.3.1.

The value of the motion threshold τ_{m} was considered to be 1% of the overall horizontal length in pixels of the SAD's frame of view, which implies that $\tau_{\mathrm{m}} \approx 9$ pixels. Moreover, the ratio γ for specifying pure horizontal or vertical motions was set to 0.2, i.e., when $|\Delta y_{t_i}| < 0.2|\Delta x_{t_i}|$, the user's motion is solely horizontal and when $|\Delta y_{t_i}| \geq 0.2|\Delta x_{t_i}|$ the secondary vertical motions of the user should be considered by the SAD. Similarly, when $|\Delta x_{t_i}| < 0.2|\Delta y_{t_i}|$, the user's motion is solely vertical and otherwise, when $|\Delta x_{t_i}| \geq 0.2|\Delta y_{t_i}|$ the secondary horizontal motions of the user should be considered by the SAD (with $\Delta x_{t_i} = x_{t_i} - x_{t_{i+1}}$ and $\Delta y_{t_i} = y_{t_i} - y_{t_{i+1}}$).

In the implementation of the motion processing algorithm, whenever at least three consecutive registers of the IPU contain invalid information, the fault tolerance algorithm was activated. Assuming an average image processing frequency of 2 Hz (see Figure 38), this implies that the user has not been detected (correctly) for at least 1.5 s. Finally, the anomaly coefficient $\tilde{C} > 1$ in (3) was set to 2.

4.7 Implementing the Standby Controller

In this section, we provide details on the implementation and tuned parameters of the standby controller explained in Section 3.4.1. Note that the displacements asked by the standby controller are executed within 0.1 sec by the SAD.

[7] https://cocodataset.org/

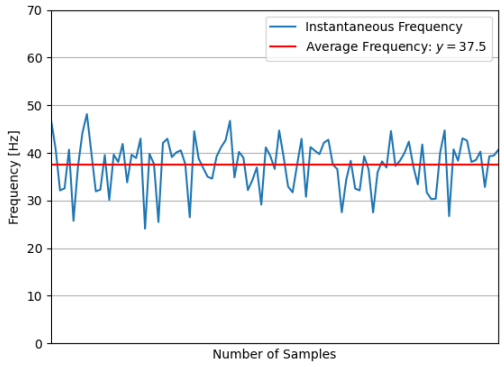


Fig. 35 Frequency of capturing pictures by the Parrot Bebop 2 camera

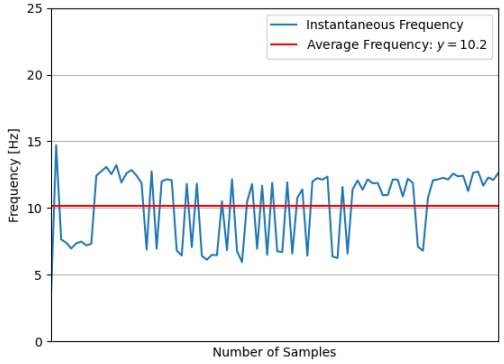


Fig. 36 Real-time performance of the IPU including ANN 1

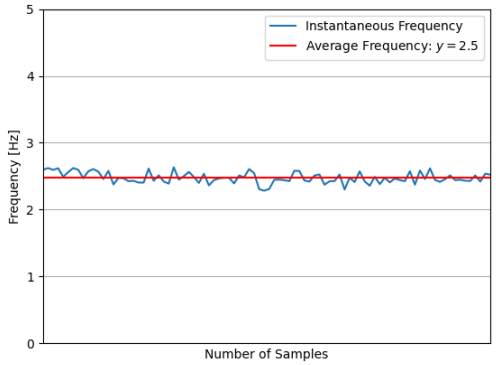


Fig. 37 Real-time performance of the IPU including ANN 2

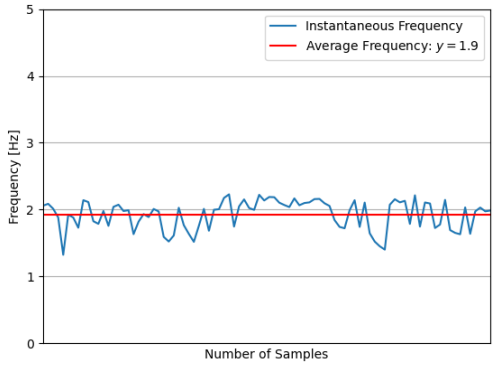


Fig. 38 Real-time performance of the IPU including both ANN 1 and ANN 2

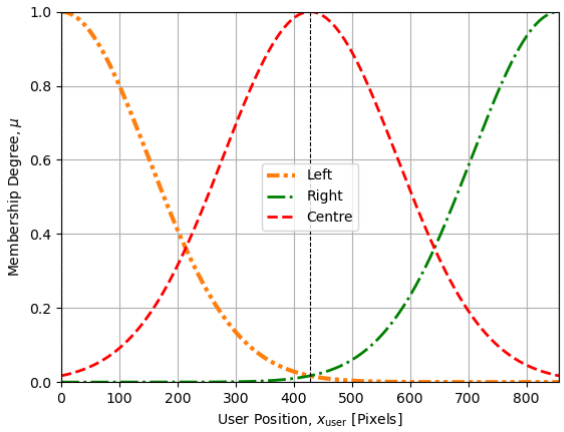


Fig. 39 Membership functions for the terms "left", "center", and "right" used by the standby controller in the case studies to center the user's image in the SAD's frame of view

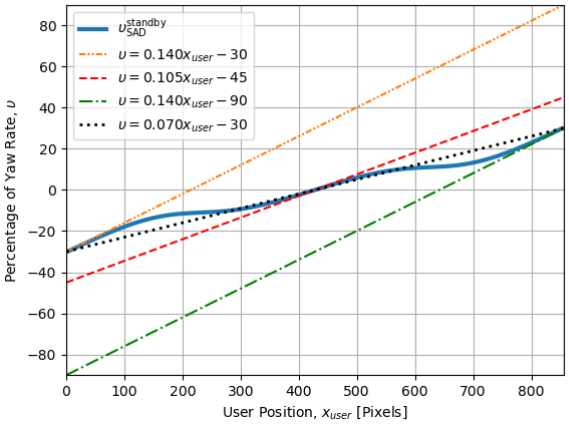


Fig. 40 Standby controller centering the user's image in the SAD's frame of view: Input-output mapping (in average) for participants in the case study

1. *Finding the user:* In our case studies in order to detect and position a user, the standby controller allows the SAD to follow a yaw rate equal to 50% of the maximum allowed yaw rate, which is 40 deg/s.
2. *Centering the user in the SAD's frame of view:* In the case studies, the threshold τ_c in (12) was set to 15 (i.e., whenever x_{user} corresponds to the center of the captured image $\pm 15\%$, the user's image is considered to be at the center of the SAD's frame of view). For the rules given by (13), the membership functions (membership functions) for the terms "left", "center", and "right" used in the case study are illustrated in Figure 39. These membership functions consist of three Gaussian functions, which are mathematically identified by their mean and standard deviation. Gaussian functions are preferred here over alternative options (e.g., triangular or trapezoidal functions) for both their smoothness and concise mathematical notations (i.e., the number of parameters that should

be tuned or personalized is reduced). These characteristics improve the adaptivity and computational efficiency of the SAD's controller in real-time applications. The mean and standard deviation of the Gaussian membership functions were selected such that these membership functions are equally spaced in the horizontal range of the image and cover all the possible input values for x_{user} (see Figure 39). Finally, the parameters a_1, a_2, a_3 (all in 1/pixels), b_1, b_2, b_3 in (13) were tuned to $a_1 = 0.140$, $a_2 = 0.105$, $a_3 = 0.140$, $b_1 = 30$, $b_2 = 45$, $b_3 = 90$.

Figure 40 shows the input-output mapping of the resulting controller corresponding to (13) for all possible values in the input domain. Figure 40 can be explained based on the influence zone of each membership function (note that the dash-double-dotted orange, dashed red, and dash-dotted green curves in Figure 40 are properly tuned outputs corresponding to, respectively, rules R1, R2, and R3 in (13)). The slopes for these curves are multiples of the slope of the line that connects points $(0, -30)$ and $(856, 30)$ (see the dotted black line in Figure 40). While the outputs corresponding to rules R1 and R3 are double of the slope of the dotted line and pass through the edge points $(0, -30)$ and $(856, 30)$, respectively, the output corresponding to rule R2 has a slope 1.5 times of that of the dotted line, and passes through $(428, 0)$, which guarantees that whenever the user is perfectly centered in the SAD's frame of view, the output of the standby controller is 0. For $x_{\text{user}} \in [0, 100]$ (where the dominant membership function is the one corresponding to "left"), the output of the standby controller (see the solid blue curve in Figure 40) shows the closest behavior to that of the membership function of "left", whereas the outputs associated with the terms "center" and "right" are more prominent when $x_{\text{user}} \in [350, 450]$ and $x_{\text{user}} \in [750, 856]$, respectively. When x_{user} is outside of these intervals, none of the rules is considered to be specifically prominent, meaning that more than one of the rules is involved in generating the output of the standby controller.

Note that whenever the input variable x_{user} reaches its minimum or maximum values the output of the standby controller is equal to -30% and 30% of the maximum yaw rate, respectively, which corresponds to the minimum overshoot when the SAD attempts to center the user's image. Moreover, whenever x_{user} is considered to be close to the center of the image, the variations in the output of the standby controller provide a trade-off between a smooth (not too sharp) motion and a reasonable speed (not less than 10% of the maximum yaw rate).

3. *Maintaining appropriate distance with the user:* To maintain the distance of the SAD with users, the value of the threshold $\tau_{\text{d},1}$ in (14) was set to 0.04 (i.e., whenever the area of the user's face-box occupies more than 4% of the total area of the captured image, the SAD is too close to the user). This value corresponds to a relative distance of almost 0.5 m between the SAD and the user, which was considered safe and convenient according to the majority of the participants in the case studies.
   The threshold $\tau_{\text{d},2}$ in (15) was set to 0.2, i.e., whenever the upper body ratio in the captured image is inferior to 20% of the overall vertical pixels, the SAD is considered to be too far from the user. This value 0.2 corresponds to a relative distance of almost 3 m between the SAD and the user. Since for larger distances, the usual motions of the user - especially the hand movements - become less ample for the SAD.
4. *Sustaining the SAD's altitude:* As it was discussed in Section 3.4.1, a default altitude of $h_{\text{SAD}}^{\text{def}} = h_{\text{user}} - 20$ (with all the magnitudes in cm) was considered for the SAD in the case studies. Moreover, for estimating the values of the thresholds τ_{h} and τ_{l}, we considered $\alpha_{\text{h}} \approx 1.375$ and $\alpha_{\text{l}} \approx 0.375$, which result in $\tau_{\text{h}} = h_{\text{user}} + 50$ and $\tau_{\text{l}} = 50$ (the values for the thresholds are given in cm).
   Since in our case studies, the variations in the vertical movement by the participants was limited, instead of considering the rules represented in (20), a simpler procedure was followed: After estimating the difference Δh_{SAD} between the latest altitude h_{SAD} registered by the SAD's sensors and the default altitude $h_{\text{SAD}}^{\text{def}}$, the standby controller executes a vertical speed of 1 m/s for Δh_{SAD} time units to position the SAD according to the default altitude.

4.8 Implementing the FL controller of Passive Game Mode 1

In order to implement the TSK-based FL controller corresponding to (21), the terms "small", "medium", and "large" for the average horizontal speed of the user's chest should mathematically be represented by fuzzy membership functions, and parameters a_9, a_{10}, a_{11}, b_9, b_{10}, and b_{11} should be tuned.

The corresponding fuzzy membership functions are illustrated in Figure 41. Note that the horizontal speed of the user's chest has been limited to a maximum of 100 pixels/s, which is determined based on several experiments performed in the Cyber Zoo with human participants. The mean values (0, 50, and 100 pixels/s) of the Gaussian membership functions were selected such that the resulting membership functions are equally spaced across the domain of the horizontal speed of the user's chest. Moreover, the standard deviation of the Gaussian membership functions were set to 20 pixels/s, since this value allows any realization within the speed domain to correspond to a

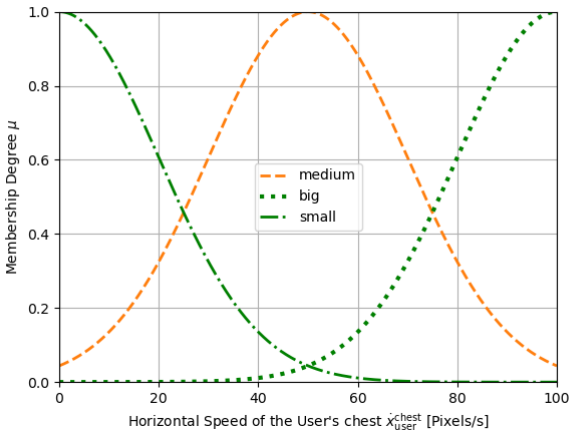


Fig. 41 Fuzzy membership functions for the terms "small", "medium", and "large" used in passive game mode 1

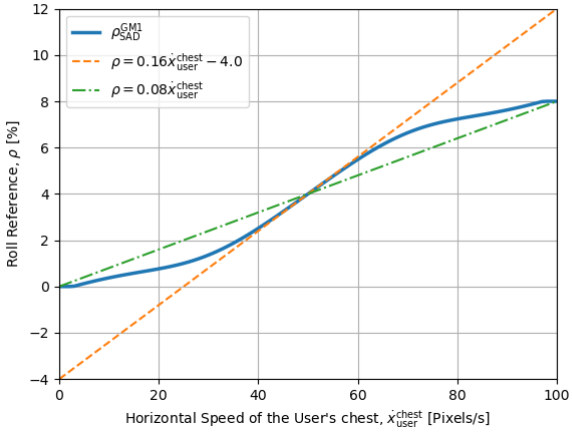


Fig. 42 TSK-based FL controller steering the SAD to implement passive game mode 1: Input-output mapping for horizontal body movements of participants

substantially high membership degree with regard to at least one of the membership functions. Moreover, the tuning parameters in (21) were set to $a_9 = -0.08$, $a_{10} = -0.16$, $a_{11} = -0.08$ (all in s/pixels), $b_9 = 0$, $b_{10} = -4$, and $b_{11} = 0$.

The resulting TSK-based FL controller for passive game mode 1 yields the input-output mapping that is illustrated in Figure 42 for the range of the input. In general, whenever the speed of the user's chest has more significant membership degrees corresponding to the sets "small" and "big" (see the dash-dotted green and dotted green curves in Figure 41) the output (see the solid blue curve in Figure 42) of the FL controller mainly approximates the behavior of a proportional controller with the gain 0.08 s/pixels (see the dash-dotted green curve in Figure 42), which ensures that the SAD never exhibits unsafe speed values. The rate of changes of the output of the controller, however, is in general slightly smaller than those of the dash-dotted green curve. The reason is to undermine the effect of unintentional movements of the user's body,which were observed in real-life experiments. Additionally, whenever the speed of the user's chest has higher membership degrees corresponding to the set "medium" (see the dashed orange curve in Figure 41), the output of the FL controller resembles the behavior of a proportional controller with a slope of 0.16 s/pixels (see the dashed orange curve in Figure 42).

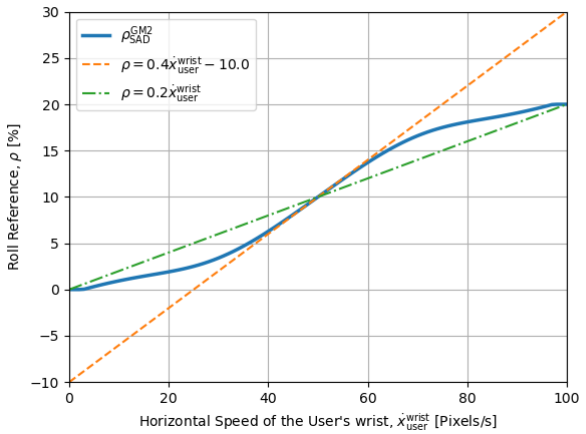


Fig. 43 TSK-based FL controller steering the SAD to implement passive game mode 2: Input-output mapping for horizontal hand movements of participants (in average) in the case study

For the proportional controller given by(22), the maximum vertical speed of the SAD and the maximum vertical speed of the user's chest are, respectively, is 1 m/s and 100 pixels/s, which correspond to a maximum value of 50% for $\zeta_{\text{SAD}}^{\text{GM1}}$. Note that the maximum vertical speed of the SAD has been selected based safety considerations for real-life experiments. Finally, the value of K^{GM1} in (22) is tuned to 0.5 s/pixels.

4.9 Implementing the FL controller of Passive Game Mode 2

To implement the TSK-based FL controller of game mode 2 given by (23), the terms "small", "medium", and "large" for the average horizontal speed of the user's wrist were defined via the same fuzzy membership functions given in Figure 41. since the horizontal displacements of the user's chest are more significant than those associated with the user's hands, the controller corresponding to game mode 2 has been tuned such that the SAD mimics a scaled version of the hand motions of the user. This way the SAD can stimulate a more engaging and entertaining interaction with the user. Consequently, the parameters of the controller given by (23)) were tuned to $a_{12} = 0.2$, $a_{13} = 0.4$, $a_{14} = 0.2$ (all in s/pixels), $b_{12} = 0$, $b_{13} = -10$, and $b_{14} = 0$.

The resulting TSK-based FL controller for passive game mode 2 yields the input-output mapping that is shown in Figure 43. Comparing Figures 42 and 43,

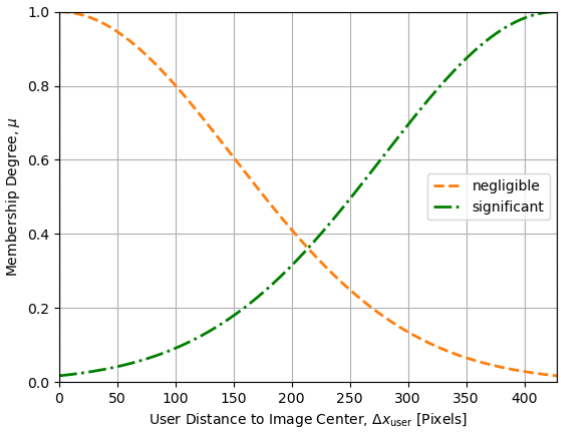


Fig. 44 Membership functions for the terms "negligible" and "significant" defined for active game modes 3 and 4

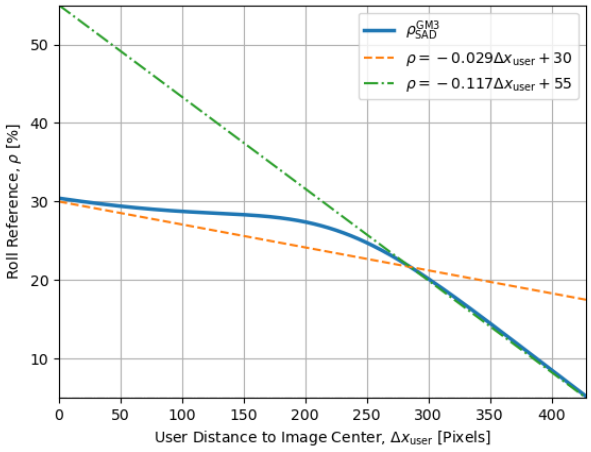


Fig. 45 TSK-based FL controller steering the SAD to implement active game mode 3: Input-output mapping for horizontal movements of the SAD

the corresponding roll rates of the SAD generated by the controller corresponding to passive game mode 2 are larger than those corresponding to the controller of passive game mode 1, which is due to the scaling factor considered for the SAD in mimicking the motions of the user's hands. For the proportional controller given by (24), the gain K^{GM2} was tuned to 0.5 s/pixels.

4.10 Implementing the FL controller of Active Game Mode 3

The fuzzy membership functions defined for the terms "negligible" and "significant" in (25) are illustrated in Figure 44. Gaussian membership functions have been considered with standard deviations of 150 pixels. Compared to passive game modes where the user initiates the movements, in active game modes extra effort is needed to assure that the user remains engaged and follows the DMT plans. Therefore, the standard deviations of the membership functions for "negligible" and "significant" are considered adaptive: For instance, when a user constantly experiences difficulties in positioning themselves in the center (according to the membership function defined for "negligible") of the SAD's frame of view and hence, receives low performance scores, the criteria for accepting Δx_{user} as *negligible* should become more lenient via increasing the standard deviation of the membership function for "negligible" and decreasing the standard deviation of the membership function for "significant". The corresponding adaptive algorithm is further discussed in Section 3.5. The tuned values for the consequent parameters of (25) include $a_{15} = -0.029$ and $a_{16} = -0.117$ (both in 1/pixels) and $b_{15} = 30$ and $b_{16} = 55$. These values guarantee that the distance travelled by the SAD in active game mode 3 is always safe and feasible (i.e., not larger than 3 m, which corresponds to a roll displacement percentage of 30%) and never becomes too small for the user to follow (i.e., not less than 0.5 m, which corresponds to a roll displacement percentage of 3%).

The resulting input-output mapping of the TSK-based FL controller is represented in Figure 45 for the range of Δx_{user}. Two distinct influence zones are distinguished, each corresponding to one of the two membership functions "negligible" and "significant". For $\Delta x_{\mathrm{user}} \leq 150$ pixels, the membership function corresponding to "negligible" (see the corresponding output in Figure 45 represented by the dashed orange curve) plays the major role in the output of the controller. For $\Delta x_{\mathrm{user}} \geq 250$ pixels, however, the membership function corresponding to "significant" is dominant (see the corresponding output represented by the dash-dotted green curve in Figure 45). Moreover, whenever Δx_{user} is roughly centered, the output of the controller does not deviate significantly from 30% of the maximum roll displacement, which guarantees safety. This is reflected in the relatively smaller slope of the dashed orange curve in Figure 45. However, whenever Δx_{user} becomes more significant (which implies a worse performances for the user), the output of the controller varies more abruptly (see the solid blue curve in Figure 45) towards the minimum role displacement equal to 5% of the maximum roll displacement.

Remark 8 Similarly to the standard deviations for the membership functions, the parameters a_{15}, b_{15}, a_{16}, and b_{16} may also be adapted in order to provide higher levels of personalization and to improve the user's engagement level. Changing these parameters indicates changing the scale of the SAD's movements, which corresponds to the movements being experienced as more or less challenging/difficult by users (see Section 3.5 for more details).

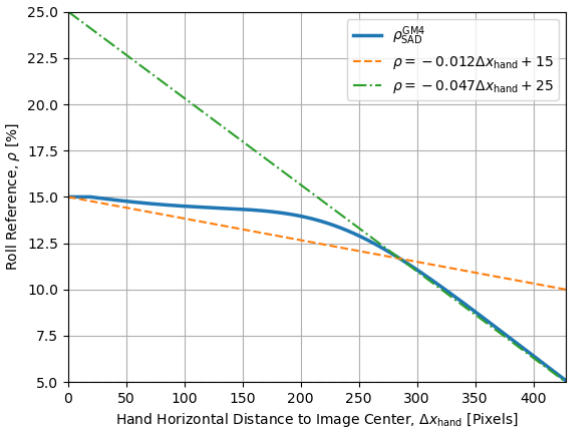


Fig. 46 TSK-based FL controller steering the SAD to implement active game mode 4: Input-output mapping for horizontal movements of the SAD

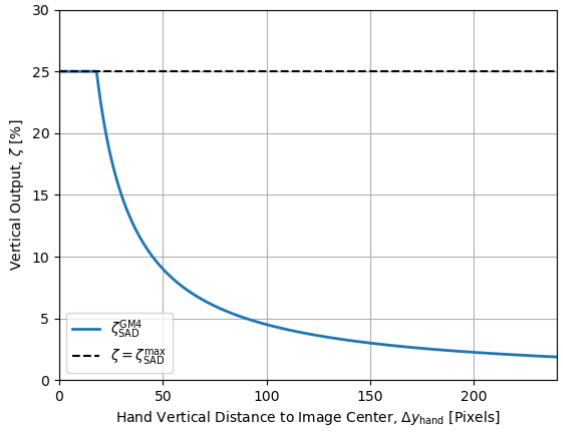


Fig. 47 Proportional controller steering the SAD to implement active game mode 4: Input-output mapping for vertical movements of the SAD

4.11 Implementing the FL controller of Active Game Mode 4

For active game mode 4, the same membership functions for the terms "negligible" and "significant" as those for game mode 3 were used (see Figure 44). Due to the larger vertical mobility of the hands compared to the chest, the proportional controller given by(27) was considered. The resulting tuned parameters are $a_{17} = -0.012$ and $a_{18} = -0.047$ (both in 1/pixels) and $b_{17} = 15$ and $b_{18} = 25$. Moreover, we set $K^{\mathrm{GM4}} = 4.5$ pixels. For safety reasons, $\zeta_{\mathrm{SAD}}^{\max} = 25\%$ was considered.

The resulting TSK-based FL controller for the horizontal movements of the SAD yields the input-output mapping given in Figure 46. Moreover, Figure 47 represents the input-output mapping of the proportional controller that steers the vertical movements of the SAD for game mode 4.

4.12 Implementing the Adaptivity and Personalization Modules

The values of the parameters that were used in the adaptivity modules for passive game modes in our experiments include: $T_{\mathrm{p}}^{\mathrm{l}} = 0.5$ and $T_{\mathrm{p}}^{\mathrm{u}} = 1.5$ for both passive game modes, $t_{\mathrm{w}}^{\mathrm{l}} = 4$ and $t_{\mathrm{w}}^{\mathrm{u}} = 10$ and $N_{\mathrm{trial}}^{\mathrm{l}} = 2$ and $N_{\mathrm{trial}}^{\mathrm{u}} = 4$ for passive game mode 1, $t_{\mathrm{w}}^{\mathrm{l}} = 4$ and $t_{\mathrm{w}}^{\mathrm{u}} = 8$ and $N_{\mathrm{trial}}^{\mathrm{l}} = 3$ and $N_{\mathrm{trial}}^{\mathrm{u}} = 8$ for passive game mode 2.

The parameters used for active game modes include: $\sigma^{\mathrm{l}} = 10$, $\sigma^{\mathrm{u}} = 400$, $\alpha_\sigma = 100$, $\beta_\sigma = -50$, $a_{15}^{\mathrm{l}} = -0.06$, $a_{15}^{\mathrm{u}} = -0.0024$, $a_{16}^{\mathrm{l}} = -0.235$, $a_{16}^{\mathrm{u}} = -0.0094$, $a_{17}^{\mathrm{l}} = -0.145$, $a_{17}^{\mathrm{u}} = -0.0058$, $a_{18}^{\mathrm{l}} = -0.585$, $a_{18}^{\mathrm{u}} = -0.0234$, $b_{15}^{\mathrm{l}} = b_{16}^{\mathrm{l}} = 20$ and $b_{15}^{\mathrm{u}} = b_{16}^{\mathrm{u}} = 80$, $b_{17}^{\mathrm{l}} = b_{18}^{\mathrm{l}} = 15$ and $b_{17}^{\mathrm{u}} = b_{18}^{\mathrm{u}} = 50$. Moreover, we have $\alpha_a = -0.4$, $\beta_a = 0.8$, $\alpha_b = 0.4$, $\beta_b = 0.8$.

For K^{GM4} in the proportional controller of active game mode 4, the following parameters were used: $\alpha^{\mathrm{GM4}} = 0.04$, $\beta^{\mathrm{GM4}} = 0.08$, $K^{\mathrm{l}} = 0.1$, and $K^{\mathrm{u}} = 1$.

For the wait time in (39) we have: $\alpha_{\mathrm{w}} = -2$ and $\beta_{\mathrm{w}} = 1$, $t_{\mathrm{w}}^{\mathrm{l}} = 4$ and $t_{\mathrm{w}}^{\mathrm{u}} = 10$ for active game mode 3, and $t_{\mathrm{w}}^{\mathrm{l}} = 2$ and $t_{\mathrm{w}}^{\mathrm{u}} = 6$ for active game mode 4.

Finally, for updating the parameter τ_{c} in (40), we have $\alpha_{\mathrm{c}} = 100$, $\beta_{\mathrm{c}} = 50$, $\tau_{\mathrm{c}}^{\mathrm{l}} = 50$, and $\tau_{\mathrm{c}}^{\mathrm{u}} = 150$.

For the personalization module, the values $\Delta w = 0.06$ and $w^{\min} = 0.05$ we considered.

5 Results and Discussion

Table 1 presents the default values considered for the personalization and adaptive parameters. Tables 2-4 represent the results corresponding to the 3 experiments executed with 10 participants. Particularly, these results include the maximum, minimum, average values, and average change (in percentage) with respect to the default values of the personalization (weights) and adaptive parameters for the four game modes after being tuned.

Regarding the weights, the average change with respect to the default values has reached values as high as 61.6% and 52.8% (for passive game mode 1 in the second and third experiments, respectively). Generally speaking, the rate of changes in the weights during the first experiment (i.e., almost the first 10 min of interactions) is less than the rate of changes in the subsequent experiments, with the largest changes occurring in the second experiment (for passive game mode 1) and in the third experiment (for passive game mode 2). This is in line with the fact that participants became more familiar and comfortable with the SAD and the setup of the experiments, and thus engaged more actively in the DMT plans. Moreover, the fact that these higher variations are observed in passive game modes (i.e., when users are expected to initiate the movement plans) is in line with the hypothesis that users felt more at ease after the first experiment to investigate more adventurous plans. Therefore, the

effect of the personalization module is expected to become more significant by adjusting the weights in the subsequent experiments, which was indeed the case. Moreover, according to these results the highest values of the tuned weights correspond to passive game mode 1. This matches the fact that all participants of these experiments were young and healthy people who found game mode 1 - which, in addition to demanding an active and initiative role from participants, requires the most physical activity amongst all the four game modes - the most joyful and engaging game mode.

One important remark to consider, however, is regarding the current simple approach for estimating the engagement level of the participants, which may by nature result in a bias in favor of passive game mode 1. Thus it will be interesting to develop a more detailed and comprehensive algorithm for estimation of the user's engagement in the future. Furthermore, the results (see particularly the minimum values of the tuned weights in Tables 2 and 3) show that the personalization algorithm may have converged too fast to the preferred game modes and has given other game modes very little chance to engage the user (see, e.g., the minimum values of the weights in Table 3, which show that within 20 min (or less) of interaction with the user, there is at least one participant per game mode for whom the weight of that particular game mode has reached the minimum weight). More specifically, for 80% of the participants the weight corresponding to at least one game mode has reached less than 0.1 and for 60% of them the weight has already reached the minimum value of 0.05 within the first 20 min. Thus, in order to avoid the convergence of the personalization algorithm prematurely and to give all game modes enough chances to engage a user, adjustments in the weight tuning algorithm are recommended. For instance, either Δw in (41) and (42) should be reduced significantly, or (41) and (42) should be reformulated.

Regarding the adaptivity module, we first discuss the results for passive game modes 1 and 2. From Tables 2-4 the wait time t_{w} has evolved continuously during the experiments. Given the high diversity of the tuned values corresponding to different participants in different experiments, t_{w} has successfully been adapted for each user in the course of different experimental conditions. Furthermore, passive game mode 1 corresponds to higher values of t_{w} compared to passive game mode 2, which can be explained based on the fact that whole body movements are slower and hence need a larger wait time than hand movements. For the number N_{trial} of trials within one episode, considering the high engagement of the participants in the two passive game modes, we expect N_{trial} to increase. The results shown in Tables 2-4 confirm this, where by the end of the third experiment N_{trial} has reached an average value of 3.8 for passive game mode 1 (where for 80% of the participants, N_{trial} has reached its upper bound, i.e., 4) and an average value of 4.8 for passive game mode 2 (where for 90% of the participants, N_{trial} has reached its upper bound, i.e., 5).

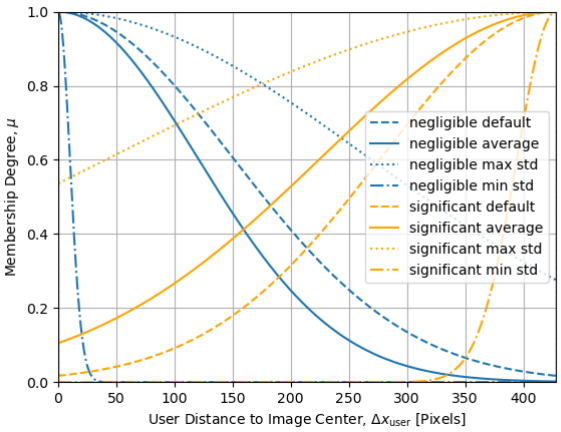


Fig. 48 Variations generated by the adaptivity module in the membership functions corresponding to active game mode 3

Regarding the performance of the adaptivity module for active game modes, we first discuss the variations in the membership functions for "negligible" and "significant" (or more specifically in σ_{neg} and σ_{sig}). Figures 48 and 49 illustrate the corresponding Gaussian membership functions used initially (see the dashed curves), as well as the Gaussian membership functions corresponding to the minimum (see the dash-dotted curves), the maximum (see the dotted curves), and the average (see the solid curves) values of σ_{neg} and σ_{sig} after being tuned by the end of the third experiment for active game modes 3 and 4.

For active game mode 3, the maximum values for the standard deviations for "negligible" and "significant" are, respectively, 383.3 and 266.7 (also see the results in Table 4), which correspond to an increase of, respectively, 155% and 78% with respect to their default values (i.e., 150 and 150). The minimum values of σ_{neg} and σ_{sig} for active game mode 3 are, respectively, 10.0 and 33.3, corresponding to a decrease of 93.3% and 77.8% from their default values. Finally, for the average values of σ_{neg} and σ_{sig} for active game mode 3, the tuned values are 119.7 and 201.7. Given the discussions in Section 3.5.2, a good performance observed from users is associated to a decrease in the standard deviation for the term "negligible" and an increase in the standard deviation for the term "significant". Taking into account that the proposed DMT plans in the experiments were not complicated for the participants and thus the majority exhibited a good performance, we expect to see the described effect in σ_{neg} and σ_{sig}, which is confirmed by the average of the tuned values of these parameters.

For active game mode 4, similarly to active game mode 3, the adaptivity module has performed excellently. The high diversity in the maximum, minimum,

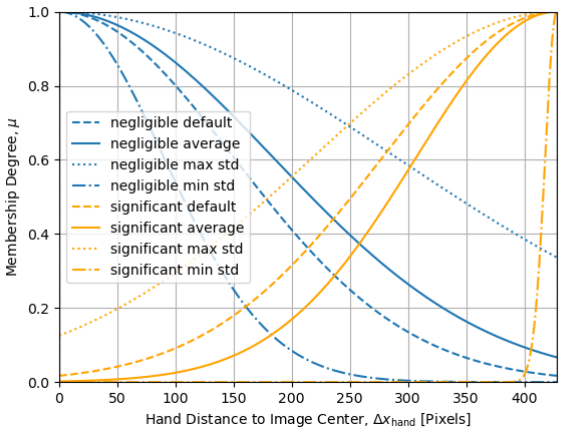


Fig. 49 Variations generated by the adaptivity module in the membership functions corresponding to active game mode 4

and average values of σ_{neg} and σ_{sig} after being tuned for various participants shows that these values have been estimated according to the variations in the experiments and users. In particular, for active game mode 4, the minimum values of the tuned parameters σ_{neg} and σ_{sig} show, respectively, a 40% and a 93% decrease with respect to their default values (150 and 150). Moreover, the maximum tuned values are 93% and 40% larger than the default values for, respectively, σ_{neg} and σ_{sig}. The main difference compared to active game mode 3, however, is that the average value σ_{neg} has increased 22.7% to 184 while the average value of σ_{sig} has decreased 19.3% to 121, which corresponds to a performance worse than that observed in active game mode 3.

Next, we discuss the changes made by the adaptivity module to the input-output mappings of the controllers corresponding to active game modes 3 and 4, where the default mappings and the average mappings after the parameters of the controllers were tuned are illustrated in Figures 50 and 51. Moreover, Figure 52 shows the input-output mapping corresponding to the vertical controller of active game mode 4 before and after tuning the controller. On average, for active game mode 3, the adaptivity module yielded an entirely different input-output mapping, whereas for active game mode 4 the resulting input-output mapping showed a similar (but fine-tuned) behavior as the original input-output mapping. As a result of tuning the parameters of the controller for active game mode 3, a poor mimicking receives a higher reward, which corresponds to displacements with a larger amplitude (see the solid orange curve in Figure 51). The reason is that this game mode is too easy for nearly all participants, and thus their performance is almost always perceived as good by the adaptivity module, resulting in a fast convergence to the thresholds considered for the tuning parameters. Additionally, Table 4 shows that 9 out of the 10 participants managed to minimize the parameter τ_{c} to its lowest bound 50. We expect to observe different results for the input-output mapping of active game mode 3 with a different group of participants with motor skill impairments or ASD.

Active game mode 4, on the contrary, was not considered to be overly simplistic for the participants. The values obtained for τ_{c} (see Table 4) after all 3 sets of experiments remained close to the default value of 100. Moreover, for none of the users the tuned value of τ_{c} reached its minimum value of 50 in active game mode 4. These results are consistent with the findings regarding the tuned membership functions for "negligible" and "significant" discussed earlier in this section. Similarly, the distribution of the tuned values of t_{w} for active game mode 3 (see Table 4) imply that, for the selected participants, this game mode was not very challenging, where the average value of t_{w} was around 4 s and 5 s, with 80% of the participants being able to minimize the value of t_{w} to the lower bound (i.e., 4 s) for this parameter. For active game mode 4, however, a much wider range of values were observed for t_{w} after being tuned.

6 Limitations and Topics for Future Work

Given the relative homogeneity among the participants (in terms of motor ability and that none of the participants had been diagnosed with ASD), increasing the number and potentially the variety of the participants to provide a more representative sample is expected to increase the fidelity of the results. In particular, real-life experiments with volunteer participants who have been diagnosed with ASD is one of the next research steps.

Furthermore, the fact that the IPU performs at a frequency of 2 Hz may pose restrictions for DMT. Such scenarios require the SAD to move according to the frequency of a melody, which may exceed the 2 Hz limit. In the experiments that were performed in the Cyber Zoo, this frequency limit was particularly restrictive for passive game mode 2, since participants had to be instructed to avoid moving their hands too fast not to exceed the 2 Hz limit.

Currently, the IPU is not able to detect physical barriers, such as walls, which may restrain the movements of the SAD due to safety considerations, especially when the SAD flies in smaller indoor spaces. Moreover, currently it is assumed that only one person is present in the images captured by the SAD. This, however, can create problems when other people (e.g., a therapist or caregiver) are present in the DMT session. Moreover, based on an analysis of the results of the experiments, developing a more sophisticated system for quantifying the engagement level of participants in various game modes can potentially improve the personalization of the SAD's control system.

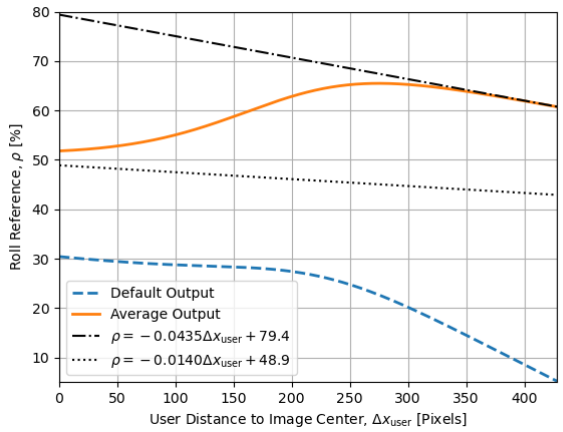


Fig. 50 Input-output (default and average) mappings for the horizontal motion controller corresponding to active game mode 3

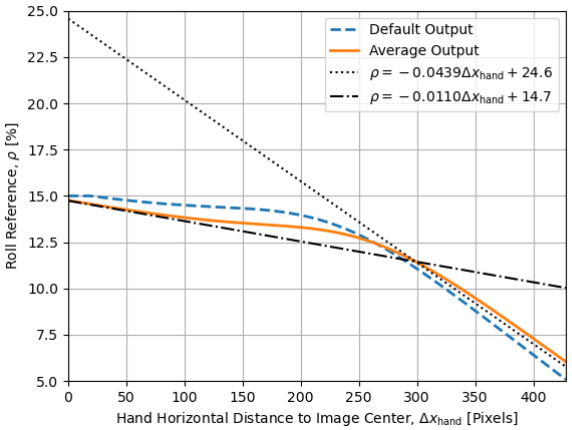


Fig. 51 Input-output (default and average) mappings for the horizontal motion controller corresponding to active game mode 4

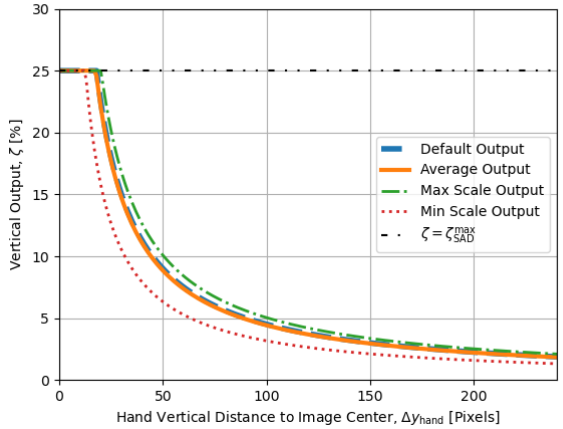


Fig. 52 Input-output (default and average) mappings for the vertical motion controller corresponding to active game mode 4

In the future, SADs are expected to be used in real DMT sessions for real-time interactions with people with ASD. Therefore, the IPU needs to be improved in the sense of both computational efficiency and detection of physical barriers. Moreover, a music analysis module should be developed and used by the SAD in order to analyze a music piece in real time and decide accordingly about the speed and frequency of the SAD's movements. Finally, a custom-made drone that further meets the requirements (regarding attractiveness of the appearance and safety) of DMT will be designed and developed to further improve the outcome of DMT sessions via SADs.

7 Conclusion

This study presents a first step towards the development of a fully functional SAD capable of performing DMT, a project that has never previously been attempted. Overall, the four game modes developed were successfully tested on a sample of 10 participants. In addition, the drone showcased the ability do adapt to each user it interacted with, both in terms of selecting the adequate game modes and adapting both the membership functions and the consequent terms of the FL controllers used, which rendered each user's experience unique. Thus, the control algorithms developed are considered to have yielded an adaptive and personalized SAD.

Although the control algorithm developed is considered successful in what concerns its adaptivity relative to the sample of participants selected, the homogeneity within the participants considered (in particular, the fact that no participant considered did not present any motor deficiency or were diagnosed with ASD) prevents making a generalizing claim regarding the efficacy of the game modes developed and the adaptive algorithm in therapy sessions aimed at social and cognitive rehabilitation of children with ASD.

Nonetheless, this study marked the first theoretical conception of a novel type of SAR: a Socially Assistive Drone, along with the first documentation of experimental test runs of the first SAD on human participants. Although not yet tested on children with ASD, the obtained results open the way for further research on control algorithms enabling SADs to perform DMT in real time, while identifying some of the major challenges and limitations which must be addressed in subsequent research. All things considered, this study represents a step forward towards the final goal of hindering the severe social and motor impairments often found in children suffering from ASD.

8 Acknowledgements

This research has been supported by the NWO - Open Mind project "Drones Interacting with Children with

Autism via Dance Movement Therapy" (18071), which has been financed by the Netherlands Organisation for Scientific Research (NWO). We would like to especially thank Dr. R. Van der Hallen, Clinical Psychologist at the Erasmus University Rotterdam, for providing extensive input on therapeutic effects of dance movement therapy for children with autism and assessment of the expected therapeutic outcomes. Moreover, we thank Dr. A. Sokolowska for training the artificial neural networks. Finally, we thank Dr. R. Samaritter for informative meetings and discussions about dance movement therapy.

Participants	Weights				Adaptive Parameters																				
					GM 1		GM 2		GM 3								GM 4								
	GM 1	GM 2	GM 3	GM 4	t_w	N_{trial}	t_w	N_{trial}	τ_{c}	t_{w}	a_{15}	a_{16}	b_{15}	b_{16}	σ_{neg}	σ_{sig}	τ_{c}	t_{w}	a_{17}	a_{18}	b_{17}	b_{18}	σ_{neg}	σ_{sig}	K^{GM4}
Default participant	0.25	0.25	0.25	0.25	6.00	3	5.00	3	64	3.0	-0.1170	-0.0292	55.0	30.0	150.0	150.0	100	3.0	-0.0467	-0.0117	25.0	15.0	150	150	0.750

Table 1 Default values of the adaptive and personalization parameters

Participants	Weights				Adaptive Parameters																				
					GM 1		GM 2		GM 3								GM 4								
	GM 1	GM 2	GM 3	GM 4	t_w	N_{trial}	t_w	N_{trial}	τ_{c}	t_{w}	a_{15}	a_{16}	b_{15}	b_{16}	σ_{neg}	σ_{sig}	τ_{c}	t_{w}	a_{17}	a_{18}	b_{17}	b_{18}	σ_{neg}	σ_{sig}	K^{GM4}
Participant 1	0.17	0.41	0.25	0.17	6.50	4	6.36	5	50	5.0	-0.0748	-0.0187	79.2	43.2	50.0	250.0	70	2.4	-0.0411	-0.0103	28.0	16.8	120	180	0.840
Participant 2	0.47	0.11	0.17	0.25	6.50	4	6.50	4	50	6.0	-0.0935	-0.0234	66.0	36.0	100.0	200.0	60	2.2	-0.0389	-0.0097	28.8	17.3	110	190	0.864
Participant 3	0.31	0.15	0.31	0.23	6.49	4	4.83	4	50	4.3	-0.0648	-0.0162	80.0	49.0	16.7	283.3	100	3.0	-0.0467	-0.0117	25.0	15.0	150	150	0.749
Participant 4	0.33	0.33	0.25	0.09	6.04	4	5.06	5	50	5.0	-0.0748	-0.0187	79.2	43.2	50.0	250.0	140	3.8	-0.0538	-0.0135	20.8	12.5	190	110	0.624
Participant 5	0.47	0.07	0.23	0.23	6.86	4	5.00	3	50	6.0	-0.0935	-0.0234	66.0	36.0	100.0	200.0	130	3.6	-0.0523	-0.0131	22.0	13.2	180	120	0.660
Participant 6	0.31	0.23	0.23	0.23	6.78	4	6.00	5	50	5.0	-0.0748	-0.0187	79.2	43.2	50.0	250.0	75	2.5	-0.0411	-0.0103	27.0	16.2	125	175	0.810
Participant 7	0.37	0.21	0.21	0.21	5.71	4	4.28	5	50	5.3	-0.0806	-0.0202	74.5	40.6	66.7	233.3	105	3.1	-0.0471	-0.0118	24.2	14.5	155	145	0.726
Participant 8	0.29	0.05	0.29	0.37	6.18	4	5.00	3	50	5.7	-0.0872	-0.0218	70.4	38.4	83.3	216.7	90	2.8	-0.0445	-0.0111	25.8	15.5	140	160	0.774
Participant 9	0.05	0.13	0.45	0.37	6.28	4	5.28	5	50	4.0	-0.0387	-0.0097	78.0	50.0	10.0	400.0	150	4.4	-0.0601	-0.0150	18.3	11.0	220	80	0.550
Participant 10	0.05	0.31	0.25	0.39	5.11	3	4.87	5	50	5.7	-0.0877	-0.0219	70.6	38.5	83.3	216.7	120	3.4	-0.0505	-0.0126	23.0	13.8	170	130	0.690
Max	0.47	0.41	0.45	0.39	6.86	4	6.50	5	50	6.0	-0.0387	-0.0097	80.0	50.0	100.0	400.0	150	4.4	-0.0389	-0.0097	28.8	17.3	220	190	0.864
Min	0.05	0.05	0.17	0.09	5.11	3	4.28	3	50	4.0	-0.0935	-0.0234	66.0	36.0	10.0	200.0	60	2.2	-0.0601	-0.0150	18.3	11.0	110	80	0.550
Average	0.28	0.20	0.26	0.25	6.24	4	5.32	4	50	5.2	-0.0770	-0.0193	74.3	41.8	61.0	250.0	104	3.1	-0.0476	-0.0119	24.3	14.6	156	144	0.729
Average change (%)	12.80	-20.00	5.60	1.60	4.08	30	6.34	46.7	-22.1	73.33	-34.1	-34.1	35.1	39.4	-59.3	66.7	4.00	4.00	1.88	1.88	-2.85	-2.84	4.0	-4.0	-2.85

Table 2 Adaptive parameters tuned in experiment 1 (t_{w} is given in seconds)

Participants	Weights				Adaptive Parameters																				
					GM 1		GM 2		GM 3								GM 4								
	GM 1	GM 2	GM 3	GM 4	t_w	N_{trial}	t_w	N_{trial}	τ_{c}	t_{w}	a_{15}	a_{16}	b_{15}	b_{16}	σ_{neg}	σ_{sig}	τ_{c}	t_{w}	a_{17}	a_{18}	b_{17}	b_{18}	σ_{neg}	σ_{sig}	K^{GM4}
Participant 1	0.55	0.15	0.15	0.15	6.50	4	6.45	5	66.7	5.3	-0.0798	-0.0199	73.9	40.3	266.7	33.3	70	2.4	-0.0411	-0.0103	28.0	16.8	120	180	0.840
Participant 2	0.63	0.05	0.11	0.21	6.82	4	5.67	5	50.0	5.3	-0.0814	-0.0204	75.1	41.0	166.7	133.3	120	3.4	-0.0485	-0.0121	22.1	13.3	250	50	0.664
Participant 3	0.49	0.09	0.25	0.17	6.97	4	4.83	5	50.0	4.0	-0.0518	-0.0130	80.0	50.0	233.3	66.7	50	2.0	-0.0373	-0.0093	30.0	18.0	100	200	0.899
Participant 4	0.65	0.25	0.05	0.05	6.87	4	5.76	5	50.0	4.3	-0.0638	-0.0160	80.0	48.4	216.7	83.3	140	3.8	-0.0538	-0.0135	20.8	12.5	190	110	0.624
Participant 5	0.41	0.05	0.23	0.31	7.69	4	5.00	3	50.0	4.0	-0.0518	-0.0130	80.0	50.0	66.7	233.3	100	3.0	-0.0444	-0.0111	23.8	14.3	90	210	0.714
Participant 6	0.67	0.11	0.11	0.11	6.01	4	6.00	5	50.0	5.0	-0.0748	-0.0187	79.2	43.2	50.0	250.0	75	2.5	-0.0411	-0.0103	27.0	16.2	125	175	0.810
Participant 7	0.69	0.05	0.17	0.09	5.45	4	4.28	5	50.0	4.0	-0.0516	-0.0129	80.0	50.0	66.7	233.3	110	3.2	-0.0434	-0.0109	26.2	15.7	125	175	0.785
Participant 8	0.25	0.07	0.29	0.39	6.33	4	5.00	5	50.0	4.0	-0.0558	-0.0140	80.0	50.0	116.7	183.3	80	2.6	-0.0414	-0.0104	26.2	15.7	150	150	0.785
Participant 9	0.05	0.05	0.59	0.31	6.28	4	5.78	5	50.0	4.0	-0.0187	-0.0047	80.0	50.0	116.7	293.3	120	3.8	-0.0430	-0.0107	21.4	12.8	70	280	0.642
Participant 10	0.05	0.39	0.17	0.39	5.11	3	5.07	5	50.0	4.0	-0.0606	-0.0151	80.0	50.0	133.3	166.7	90	2.8	-0.0437	-0.0109	25.4	15.3	100	200	0.763
Max	0.69	0.55	0.59	0.39	7.69	4	6.45	5	66.7	5.3	-0.0187	-0.0047	80.0	50.0	266.7	293.3	140	3.8	-0.0373	-0.0093	30.0	18.0	250	280	0.899
Min	0.05	0.05	0.05	0.05	5.11	3	4.28	3	50.0	4.0	-0.0814	-0.0204	73.9	40.3	50.0	33.3	50	2.0	-0.0538	-0.0135	20.8	12.5	70	50	0.624
Average	0.40	0.17	0.21	0.22	6.40	4	5.38	5	51.7	4.4	-0.0590	-0.0148	78.8	47.3	143.3	167.7	96	3.0	-0.0438	-0.0109	25.1	15.1	132	173	0.753
Average change (%)	61.60	-33.60	-15.20	-12.80	6.70	30	7.66	60	-19.5	46.67	-49.5	-49.5	43.3	57.6	-4.4	11.8	-4.50	-1.67	-6.31	-6.31	0.34	0.34	-12.0	15.3	0.340

Table 3 Adaptive parameters tuned in experiment 2 (t_{w} is given in seconds)

Participants	Weights				Adaptive Parameters																				
					GM 1		GM 2		GM 3								GM 4								
	GM 1	GM 2	GM 3	GM 4	t_w	N_{trial}	t_w	N_{trial}	τ_{c}	t_{w}	a_{15}	a_{16}	b_{15}	b_{16}	σ_{neg}	σ_{sig}	τ_{c}	t_{w}	a_{17}	a_{18}	b_{17}	b_{18}	σ_{neg}	σ_{sig}	K^{GM4}
Participant 1	0.13	0.77	0.05	0.05	6.84	4	6.28	5	66.7	5.3	-0.0798	-0.0199	73.9	40.3	266.7	33.3	70	2.4	-0.0411	-0.0103	28.0	16.8	120	180	0.840
Participant 2	0.55	0.17	0.09	0.19	6.98	4	6.67	5	50.0	4.0	-0.0521	-0.0130	80.0	50.0	166.7	133.3	100	3.0	-0.0444	-0.0111	23.8	14.3	290	10	0.713
Participant 3	0.55	0.05	0.23	0.17	6.96	4	4.78	5	50.0	4.0	-0.0265	-0.0066	80.0	50.0	10.0	383.3	100	3.0	-0.0449	-0.0112	24.1	14.5	250	50	0.724
Participant 4	0.27	0.53	0.15	0.05	8.37	3	5.50	5	50.0	4.0	-0.0595	-0.0149	80.0	50.0	66.7	233.3	150	4.6	-0.0627	-0.0157	17.6	10.5	210	90	0.527
Participant 5	0.47	0.05	0.29	0.19	8.19	4	5.00	3	50.0	4.0	-0.0419	-0.0419	80.0	50.0	183.3	116.7	100	3.0	-0.0444	-0.0111	23.8	14.3	90	210	0.714
Participant 6	0.75	0.15	0.05	0.05	6.49	4	5.56	5	50.0	4.3	-0.0648	-0.0162	80.0	49.0	216.7	83.3	75	2.5	-0.0411	-0.0103	27.0	16.2	125	175	0.810
Participant 7	0.83	0.05	0.07	0.05	6.25	4	4.28	5	50.0	4.0	-0.0330	-0.0083	80.0	50.0	133.3	166.7	120	3.4	-0.0451	-0.0113	25.1	15.1	185	115	0.754
Participant 8	0.17	0.09	0.33	0.41	5.92	4	5.08	5	50.0	4.0	-0.0247	-0.0062	80.0	50.0	10.0	300.0	80	2.6	-0.0392	-0.0098	24.8	14.9	150	150	0.743
Participant 9	0.05	0.05	0.53	0.37	6.44	4	6.28	5	50.0	4.0	-0.0187	-0.0047	80.0	50.0	110.0	300.0	70	2.8	-0.0336	-0.0084	25.0	15.0	230	120	0.751
Participant 10	0.05	0.49	0.21	0.25	5.11	3	5.02	5	50.0	4.0	-0.0338	-0.0084	80.0	50.0	33.3	266.7	80	2.6	-0.0419	-0.0105	26.5	15.9	190	110	0.794
Max	0.83	0.77	0.53	0.41	8.37	4	6.67	5	66.7	5.3	-0.0187	-0.0047	80.0	50.0	266.7	383.3	150	4.6	-0.0336	-0.0084	28.0	16.8	290	210	0.840
Min	0.05	0.05	0.05	0.05	5.11	3	4.28	3	50.0	4.0	-0.0800	-0.04	73.90	40.30	10.00	33.30	70.00	2.40	-0.06	-0.02	17.60	10.50	90.00	10.00	0.53
Average	0.38	0.24	0.20	0.18	6.75	3.8	5.44	4.8	51.7	4.2	-0.0440	-0.0100	79.4	48.9	119.7	201.7	94.50	2.99	-0.044	-0.01	24.57	14.74	184.00	121.00	0.74
Average change (%)	52.80	-4.00	-20.00	-28.80	12.60	26.70	8.86	60	-19.50	38.89	-62.80	-52.00	44.30	63.10	-20.20	34.40	-5.50	-0.33	-6.15	-6.15	-1.73	-1.73	22.70	-19.30	-1.73

Table 4 Adaptive parameters tuned in experiment 3 (t_{w} is given in seconds)

Appendices

A Quadcopter Dynamics

In this appendix, approximate relationships are derived between the roll angle ϕ and the pitch angle angle θ, and the corresponding lateral displacement ΔX and longitudinal displacement ΔZ. The derived approximate relationships are considered to be accurate enough for the purposes of this paper.

In order to distinguish the inertial and body reference frames, the uppercase notation XYZ is used when referring to the inertial reference frame, whereas lowercase xyz refers to the body reference frame.

In the game modes developed for the SAD detailed in the main paper, the control input selected either modifies the pitch angle θ or the roll angle ϕ, never both simultaneously. This enables addressing the rigid body dynamics of the drone by analysing the rotations around the X and Z axis independently and thus highly simplifies the problem at hand, particularly when compared to alternative approaches which consider coupled dynamics and rotation matrices [29], which are considered to leave the scope of this research.

The SAD's force diagrams for generic lateral and longitudinal motions are represented in Figures 53 and 54, respectively.

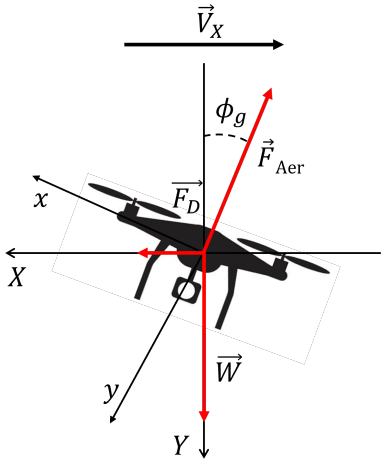


Fig. 53 Force diagram for lateral motion to the right

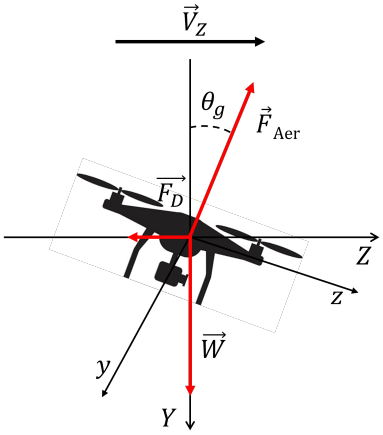


Fig. 54 Force diagram for longitudinal motion forward

where $\vec{V}_X$ represents lateral velocity, $\vec{V}_Z$ refers to longitudinal velocity, $\vec{F}_D$ the drag force, $\vec{F}_{Aer}$ the aerodynamic force resulting from the propellers and $\vec{W}$ the quad-copter's weight. The intensity of the previous forces is denoted as F_D, F_{Aer} and W, whereas the total velocity module in the inertial frame is represented as V. In addition, θ_g and ϕ_g are hereby defined as generic pitch and roll angles, respectively.

As described in [29] the drag equation from fluid dynamics defines the frictional force $\vec{F}_D$ as:

$$F_D = \frac{1}{2}\rho C_d A V^2 \tag{43}$$

where A represents reference area, C_d the drag coefficient and ρ the air density.

From the previous equation, it follows that the drag force increases as the drone gains speed in either the lateral or longitudinal dimensions. Consequently, for a given generic roll (in case of a lateral motion) or pitch (in case of longitudinal motion) angles, a horizontal state of equilibrium will be achieved between the horizontal component of $\vec{F}_{Aer}$ and $\vec{F}_D$ in the direction of motion (X and Z respectively) as the drone's velocity increases. When considering the vertical direction, given that the motions being analysed are strictly contained within a horizontal plane, the drone always remains in equilibrium.

By analysing the sum of forces in the vertical and horizontal directions in the inertial reference frame after equilibrium is achieved for a lateral motion, we get:

$$\begin{cases} \sum \vec{F}_Y = 0 \\ \sum \vec{F}_X = 0 \end{cases} \tag{44}$$

in the vertical direction, we obtain:

$$F_{Aer}\cos(\phi_g) = W \Leftrightarrow F_{Aer} = \frac{W}{|\cos(\phi_g)|} \tag{45}$$

Which can be replaced in the second equation, yielding:

$$F_{Aer}|\sin(\phi_g)| = F_D \Leftrightarrow W|\tan(\phi_g)| = \frac{C_D A \rho (V_X^{\inf})^2}{2} \Leftrightarrow V_X^{\inf} = \sqrt{\frac{2W}{C_{d_x} A \rho}}\sqrt{|\tan(\phi_g)|} \tag{46}$$

where $V_X^{\inf}$ denotes the lateral speed at equilibrium or terminal lateral speed and C_{d_x} is the drag coefficient in the X direction.

When applying the same procedure considering a longitudinal motion, we obtain:

$$V_Z^{\inf} = \sqrt{\frac{2W}{C_{d_Z} A \rho}} \sqrt{|\tan(\theta_g)|} \tag{47}$$

with $V_X^{\inf}$ denoting the terminal longitudinal speed and C_{d_x} representing the drag coefficient in the Z direction.

Furthermore, given that all the values in the previous equation except ϕ_g and θ_g are constant, they can be grouped in two aerodynamic constants denoted as K_{aer_X} and K_{aer_Z}, defined as:

$$K_{aer_X} = \sqrt{\frac{2W}{C_{d_X} A \rho}} \tag{48}$$

$$K_{aer_Z} = \sqrt{\frac{2W}{C_{d_Z} A \rho}} \tag{49}$$

Consequently, the terminal speeds can be written as:

$$V_X^{\inf} = K_{aer_X} \sqrt{|\tan(\phi_g)|} \tag{50}$$

$$V_Z^{\inf} = K_{aer_Z} \sqrt{|\tan(\theta_g)|} \tag{51}$$

Since the reference angles in nearly all situations mentioned in the main article are relatively small, i.e., do not exceed 10 degrees, the small angle approximation, in other words, the first degree Taylor series expansion expressed in 52 can be used on the tangent function, yielding the final expression to obtain the terminal speeds $V_X^{\inf}$ and $V_Z^{\inf}$ from generic pitch and roll angles:

$$\tan(x) \approx x \tag{52}$$

$$V_X^{\inf} = K_{aer_X} \sqrt{|\phi_g|} \tag{53}$$

$$V_Z^{\inf} = K_{aer_Z} \sqrt{|\theta_g|} \tag{54}$$

As a final consideration, it should be noted that there are two main assumptions which underlie the use of the previous formulas for approximating the lateral and longitudinal displacement associated with a given roll or pitch angle.

The first of the two consists in the assumption that the control inputs are followed nearly instantly. This is considered to be a good approximation due to two reasons: Firstly, as was previously mentioned, due to the fact that in all the game modes developed both the roll and the pitch angles typically never exceed 10 degrees (i.e., the maximum output does not reach 50%), making them particularly small angles and, secondly, due to the magnitude of the maximum pitch and roll rates. These are set to 200 degrees/s, which means that attaining the previously mentioned small angles is expected to take less than 0.1 seconds, even if the maximum rates are never reached.

The second assumption underling the previous expressions consists in considering that the lateral and longitudinal equilibria are quickly (quasi-instantaneously) reached once the drone achieves the reference roll or pitch angles. Once more, this second assumption is based on the fact that only small angles are used as reference. This, in turn, dictates that the drag force to needed to attain horizontal equilibrium is significantly reduced (as the aerodynamic force in the direction of motion is extremely small) and thus can be quickly achieved, for low lateral and longitudinal velocities. This second assumption is fundamental for being able to link a given pitch/roll angle to a fixed longitudinal/lateral speed, according to the expressions presented for $V_X^{\inf}$ and $V_Z^{\inf}$.

Overall, according to classic control theory, the previous two assumptions dictate that the transient response before achieving steady state is quite fast for small reference roll and pitch angles, thus allowing to correspond angles directly to constant speeds (instead of accelerations) during a manoeuvre. Consequently, the lateral and longitudinal displacements of a given manoeuvre can be written according to:

$$\Delta X = V_X^{\inf} t_m \Leftrightarrow \Delta X = K_{aer_X} \sqrt{|\phi_g|} t_m \tag{55}$$

$$\Delta Z = V_Z^{\inf} t_m \Leftrightarrow \Delta Z = K_{aer_Z} \sqrt{|\theta_g|} t_m \tag{56}$$

where t_m is the manoeuvre duration, in seconds.

References

1. D. J. Ricks and M. B. Colton, "Trends and considerations in robot-assisted autism therapy," in *2010 IEEE international conference on robotics and automation*, pp. 4354–4359, 2010.
2. S. Kopp, E. Beckung, and C. Gillberg, "Developmental coordination disorder and other motor control problems in girls with autism spectrum disorder and/or attention-deficit/hyperactivity disorder," *Research in developmental disabilities*, vol. 31, no. 2, pp. 350–361, 2010.
3. H. Dadgar, J. A. Rad, Z. Soleymani, A. Khorammi, J. McCleery, and S. Maroufizadeh, "The relationship between motor, imitation, and early social communication skills in children with autism," *Iranian journal of psychiatry*, vol. 12, no. 4, p. 236, 2017.
4. M. Dziuk, J. G. Larson, A. Apostu, E. Mahone, M. Denckla, and S. Mostofsky, "Dyspraxia in autism: association with motor, social, and communicative deficits," *Developmental Medicine & Child Neurology*, vol. 49, no. 10, pp. 734–739, 2007.
5. S. Goldman, C. Wang, M. W. Salgado, P. E. Greene, M. Kim, and I. Rapin, "Motor stereotypies in children with autism and other developmental disorders," *Developmental Medicine & Child Neurology*, vol. 51, no. 1, pp. 30–38, 2009.
6. T. Higashionna, R. Iwanaga, A. Tokunaga, A. Nakai, K. Tanaka, H. Nakane, and G. Tanaka, "Relationship between motor coordination, cognitive abilities, and academic achievement in japanese children with neurodevelopmental disorders," *Hong Kong Journal of Occupational Therapy*, vol. 30, no. 1, pp. 49–55, 2017.
7. A. Cummins, J. P. Piek, and M. J. Dyck, "Motor coordination, empathy, and social behaviour in school-aged children," *Developmental Medicine & Child Neurology*, vol. 47, no. 7, pp. 437–442, 2005.
8. M. Martin, "Moving on the spectrum: Dance/movement therapy as a potential early intervention tool for children with autism spectrum disorders," *The Arts in Psychotherapy*, vol. 41, no. 5, pp. 545–553, 2014.
9. H. Takahashi, K. Matsushima, and T. Kato, "The effectiveness of dance/movement therapy interventions for autism spectrum disorder: A systematic review," *American Journal of Dance Therapy*, vol. 41, no. 1, pp. 55–74, 2019.
10. T. Fong, I. Nourbakhsh, and K. Dautenhahn, "A survey of socially interactive robots," *Robotics and autonomous systems*, vol. 42, no. 3-4, pp. 143–166, 2003.
11. B. Scassellati, H. Admoni, and M. Matarić, "Robots for use in autism research," *Annual review of biomedical engineering*, vol. 14, pp. 275–294, 2012.
12. A. Taheri, M. Alemi, A. Meghdari, H. PourEtemad, and N. M. Basiri, "Social robots as assistants for autism therapy in iran: Research in progress," in *2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, pp. 760–766, IEEE, 2014.
13. J.-J. Cabibihan, H. Javed, M. Ang, and S. M. Aljunied, "Why robots? a survey on the roles and benefits of social robots in the therapy of children with autism," *International journal of social robotics*, vol. 5, no. 4, pp. 593–618, 2013.
14. J. Kajopoulos, A. H. Y. Wong, A. W. C. Yuen, T. A. Dung, T. Y. Kee, and A. Wykowska, "Robot-assisted training of joint attention skills in children diagnosed with autism," in *International conference on social robotics*, pp. 296–305, Springer, 2015.
15. L. I. Ismail, T. Verhoeven, J. Dambre, and F. Wyffels, "Leveraging robotics research for children with autism: a review," *International Journal of Social Robotics*, vol. 11, no. 3, pp. 389–410, 2019.
16. P. Marti, M. Bacigalupo, L. Giusti, C. Mennecozzi, and T. Shibata, "Socially assistive robotics in the treatment of behavioural and psychological symptoms of dementia," in *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006. BioRob 2006.*, (Pisa, Italy), pp. 483–488, 2006.
17. S. Shamsuddin, H. Yussof, L. Ismail, F. A. Hanapiah, S. Mohamed, H. A. Piah, and N. I. Zahari, "Initial response of autistic children in human-robot interaction therapy with humanoid robot NAO," in *2012 IEEE 8th International Colloquium on Signal Processing and its Applications*, pp. 188–193, 2012.
18. L. Boccanfuso and J. M. O'Kane, "Charlie: An adaptive robot design with hand and face tracking for use in autism therapy," *International journal of social robotics*, vol. 3, no. 4, pp. 337–347, 2011.
19. C. Moro, G. Nejat, and A. Mihailidis, "Learning and personalizing socially assistive robot behaviors to aid with activities of daily living," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 7, no. 2, pp. 1–25, 2018.
20. N. Giullian, D. Ricks, A. Atherton, M. Colton, M. Goodrich, and B. Brinton, "Detailed requirements for robots in autism therapy," in *2010 IEEE International Conference on Systems, Man and Cybernetics*, (Istambul, Turkey), pp. 2595–2602, 2010.
21. E. H. Mamdani, "Advances in the linguistic synthesis of fuzzy controllers," *International Journal of Man-Machine Studies*, vol. 8, no. 6, pp. 669–678, 1976.
22. L. A. Zadeh, "Information and control," *Fuzzy sets*, vol. 8, no. 3, pp. 338–353, 1965.
23. T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, no. 1, pp. 116–132, 1985.
24. Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
25. T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *CVPR*, 2017.
26. J. Rada-Vilela, "The fuzzylite libraries for fuzzy logic control," 2018.
27. P. Developers, "Olympe," 2016. Last Time Accessed: 06-06-2021.
28. A. McGovern, "Pyparrot," 2017. Last Time Accessed: 27-02-2021.
29. D. Gheorghiţă, I. Vîntu, L. Mirea, and C. Brăescu, "Quadcopter control system," in *2015 19th International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 421–426, IEEE, 2015.

II

Literature Survey[1]

[1]Literature Survey was assessed as part of the AE4020 Literature Study course

1 Introduction

1.1. Motivation

In recent years, Autism Spectrum Disorder, commonly referred to by its initials, ASD, has increasingly gathered the interest of the scientific community, with the amount of diagnosed cases and subsequent related research increasing over the same period of time [22]. In fact, the *World Health Organization* estimates that this condition affects one in every one hundred sixty children on a global scale, with its prevalence growing by the year [27][12]. This estimate, however, is naturally tied to the amount invested in means to diagnose the disease, with highly developed countries such as the United States reaching a much more worrying number of 1.85%, that is, one in every fifty four children as depicted in Figure 1.1 [19].

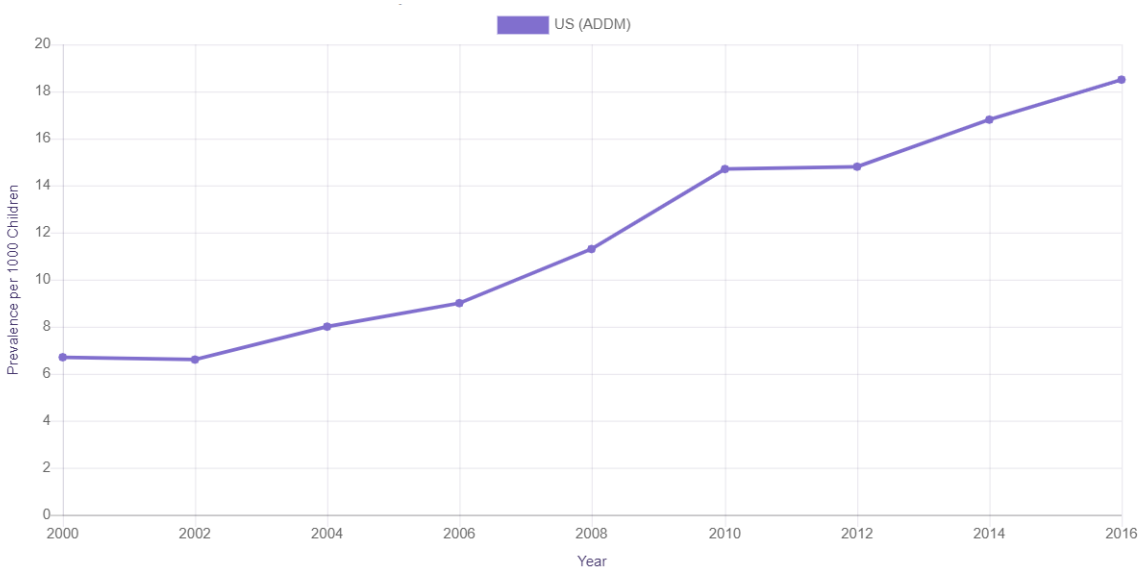


Figure 1.1: Autism and Developmental Disabilities Monitoring (ADDM) Network estimates for overall ASD prevalence in US over time [19]

One of the most fundamental aspects about how ASD is that this condition is far from a uniform disease with a constant set of identical impairments. Instead, the impairment level can be considered volatile, depending on the considered patient. Nonetheless, some conditions are present in all degrees of the spectrum. It is the case of impairments such as deficient social interactions, poor communication skills and abnormal play patterns. In fact, the word autism finds its roots in the Greek word ”autos” meaning self, referring to the sense of an isolated self, removed from social interactions. Consequently, children with severe autism may exhibit all of the previously mentioned symptoms and appear to live

in their own world, whereas higher-functioning children on the autism spectrum exhibit mild forms of these attributes and can achieve a certain level of sociability. [30]

In light of the previous statements, it is by no means an overstatement to consider ASD a condition with a considerable negative social impact, due to both its associated impairments and its frequency of diagnosis.

From an economic point of view, the scenario does not look much brighter, as the impacts associated with ASD are also far from negligible, with recent findings reporting several concerning values worldwide. In the United States, values such as a 56% disparity in salaries between mothers of children with ASD and mothers whose children have no health limitations were found, corresponding to a USD 14755 difference per year on average [7]. Additionally, the same report concludes that children suffering from ASD are 9% less likely to have both parents working, which is responsible for a 28% decrease in family earnings. In the United Kingdom, a similar scenario occurs, as the lifetime cost, after discounting, for someone with ASD and intellectual disability is estimated to reach £1.23 million, and, for someone with ASD without intellectual disability, a value of £0.80 million is estimated [18]. Furthermore, the costs of supporting children with ASD were estimated to reach £2.7 billion a year, whereas the same number relative to adults reaches the astounding value of £25 billion each year. Lastly, in Ireland it is estimated that, on average, the annual cost related to private autism spectrum disorder services, lost income and informal care per child for families reaches €28,464.89 which more than doubles the annual state expenditure per child on ASD related to health, social and educational resources of €14,192 [33].

In order to address the social and economic consequences related to ASD, therapeutic approaches have mostly been regarded as the best available tool, especially when applied as early as the condition is diagnosed, which can happen as soon as when a child is three years old [30]. In fact, early intervention has been shown to greatly increase the long-term benefits of clinical therapy in children with autism and consequently, there is a focus on diagnosing autism at earlier stages in development, which could lead to higher functionality later in life. [30].

Within the available therapy methods for autism, Dance Movement Therapy (DMT) has spiked the interests of many researchers since it was first introduced, in the 1970s. According to H. Takahashi et al. [38] DMT can be defined as ”a psychotherapeutic healing tool premised on the theory that the body and mind are connected. Bodily movements such as dancing are therapeutic because they promote mental and physical health, and have been found to improve self-esteem, quality of life, well-being, coping skills, and feelings of joy. DMT also supports the development of physical strength, as well as social and communication skills”. In their paper, the authors review the usage of DMT over the past years, emphasizing that mirroring interventions in particular have helped to enhance the social skills of participants with ASD in a wide range of ages and intellectual capabilities. Furthermore, these findings are coherent with previous research, which also indicated (although not in such a systematic way) that DMT interventions can be utilized to address this developmental connection in infants and young children with, or at high risk for, ASD [22].

Nevertheless, current therapy methods, including DMT, face several challenges, most of these related to the need of a high degree of personalization and how to respond to appropriately adapt to each child’s individual needs. This fact led to the creation and development of the concept of socially assistive robots (SARs) as an additional therapeutic tool, which will be thoroughly explained in Chapter 2.

1.2. Project Objective

Taking all that was mentioned in Section 1.1 into account, the aim of the research conducted consists on developing and implementing novel approaches for implementing personalized DMT to prevent or diminish communication impairments associated to ASD. These approaches rely on the concept of socially assistive robots, more specifically Socially Assistive Drones (SADs). In particular, we will focus on developing the decision making system, which influences the drone’s behaviour, i.e. the drone’s control module, with the ultimate goal being finding the best approach for a decision-making module that is both adaptive and effective.

In this sense, the project is closely linked to the concept of socially assistive robots meant to be used in ASD therapy. The implementations and development will involve a Parrot Bebop 2 drone, which is introduced in Chapter 2 (Figure 2.8 and Figure 2.9).

In order to reach the proposed goal, research is conducted to address a predefined set of research questions, which are listed below in Table 1.1.

Research Questions
• What is a socially assistive robot?
• Which traits or characteristics are linked to a good therapeutic performance for ASD?
• What kinds of socially assistive robots have been developed so far?
• Can/Have drones be/been used as socially assistive robots?
• Which of the mentioned traits are mainly influenced by the decision module/control system of SARs?
• Which control strategy or algorithm can better achieve the desired characteristics?
• Can several control strategies be combined in order to achieve a better trade-off?
• Which control approaches can provide decisions that eventually satisfy the desired performance criteria?

Table 1.1: Listing the Research Questions

The previously enumerated research questions have been subsequently used to implement a project plan, showcased in Table 1.2

Consequently, the following sections will be structured in a way to first answer the research questions and follow the stipulated research plan.

In this sense, Chapter 2 discusses socially assistive robots, that is, aside from introducing this concept, this chapter explores the scope of socially assistive robots that have been developed up to this point, as well as the main traits related to a successful implementation in therapy sessions. Naturally, the applications and main advantages and disadvantages related to drone usage in specific are also mentioned.

Then, Chapter 3 focuses on the decision module analysis by introducing several technical concepts related to control theory, such as fuzzy logic control, adaptive control, deep reinforcement learning.

The subsequent Chapter 4, on the other hand, focuses on the pragmatic side of the techniques described in depth in Chapter 3. In other words, it will consider the pros and cons of each of the previously presented approaches, analysing these according to the performance needed for application into therapeutic drones.

The preliminary results obtained thus far are presented in Chapter 5, together with a description of the experimental setup, software used and procedure. Furthermore, the limitations and problems identified are disclosed, along with future experimental work.

Lastly, Chapter 6 briefly summarizes the main conclusions attained in the research process and

Project Objectives
Step 1: Gather information by answering all the research questions;
Step 2: Develop a new controller based on fuzzy logic;
Step 3: Implement the beta version of developed controller in python;
Step 4: Test the beta version via real-life experiments simulated for predefined scenarios;
Step 5: Improve controller based on step 4 results by adding an adaptive module based on reinforcement learning;
Step 6: Test the final version via real-life experiments simulated for predefined scenarios.

Table 1.2: Listing the Project Objectives

mentions future work which will be conducted at a later stage of the project.

2

Socially Assistive Robots

The following section focuses on defining the concept of socially assistive robots, together with describing the main benefits related to their implementation in therapeutic applications. Furthermore, several examples from recent literature related to the development of SARs meant to be used in a variety of medical conditions, in particular ASD, are analysed and discussed in detail. Lastly, the case study of using a Parrot Bebop drone as a SAR is presented together with its inherent benefits and drawbacks.

2.1. Introducing Socially Assistive Robots

Socially assistive robots, or SARs in short, are a relatively new concept. In fact, the field of socially assistive robotics was first defined in 2005 by David Feil-Seifer et al. [13] as the intersection of Assistive Robotics and Socially Interactive Robotics (SIR).

With that said, an assistive robot is defined as a robot that gives aid or support to a human user. Consequently, research into assistive robotics includes rehabilitation robots, wheelchair robots and other mobility aides, companion robots, manipulator arms for the physically disabled, and educational robots [13]. Furthermore, these robots are intended for use in a range of environments including schools, hospitals, and homes. In contrast, SIRs were first described by Fong in [14] as those whose main task is some form of social interaction. According to the author, the previous term was introduced to distinguish social interaction from teleoperation in human robot interaction.

Taking the previous two concepts into account, it is considered that socially assistive robots share with assistive robots the goal to provide assistance to human users, but it specifies that the assistance is through social interaction. Because of the emphasis on social interaction, SARs have a similar focus to SIRs. Regarding SIRs, the robot's goal is to develop close and effective interactions with the human for the sake of interaction itself. In contrast, when a SAR is concerned, the robot's goal is to create close and effective interaction with a human user for the purpose of giving assistance and achieving measurable progress in fields such as convalescence, rehabilitation or learning. [13][25]

Ever since its introduction in the scientific community, the concept of SARs has matured and expanded, with several examples of SARs having been successfully developed for therapeutic applications related to a wide range of medical conditions, as will be described later on, in Section 2.3.

2.2. Main Characteristics Required in SARs

The recent growth in research related to SARs has led researchers to develop a wide variety of robots, each presenting its own set of traits adapted to the therapeutic application being considered. Out of this set of traits, elegantly described in several articles [15] [30], five were considered crucial for a successful therapeutic outcome associated with ASD: appearance, adaptability, simplicity, responsiveness and, lastly, autonomy. Due to their high impact in developing and adequate SAR, which remains the final goal of the thesis project, these concepts are introduced and described below.

2.2.1. Appearance

The first trait mentioned is perhaps the most straightforward one: when designing a SAR for a specific purpose, its design is paramount. Consequently, depending on which condition the therapy is meant to address, a significant amount of designs have been used on SARs through the years.

To illustrate the previous point, in *Trends and Considerations in Robot-Assisted Autism Theory*, Daniel Ricks et al. in [30] proposed a scale to compare robot types used in autism therapy research. This scale includes five different classes: Android, Mascot, Mechanical, Animal and Non-Humanoid Mobile Robot, with each one of them being described in Figure 2.1.

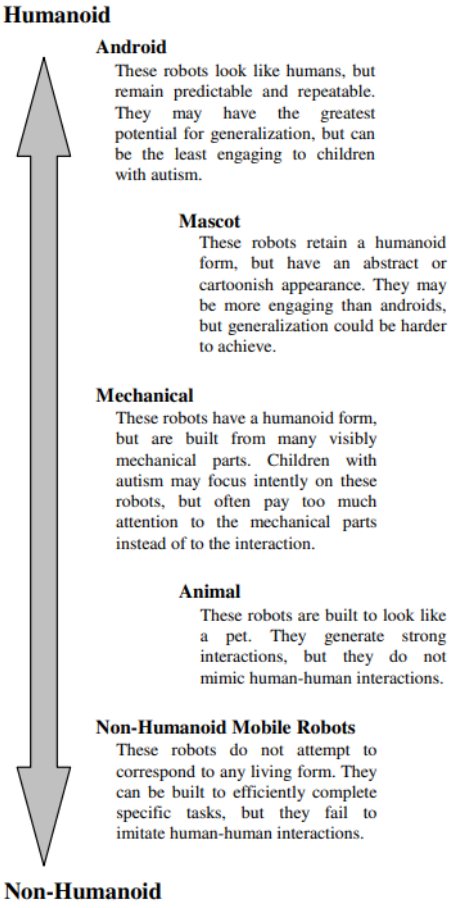


Figure 2.1: SAR classification scale proposed by Daniel Ricks et al. [30]

In the previous scale, the criteria used to distinguish classes is none other than appearance. In this sense, SARs who look similar to a human figure are placed in the categories closer to the humanoid end of the spectrum, such as Android or Mascot, whereas those that present few human features can be found closer to the non-humanoid classes such as Non-Humanoid Mobile Robots, where drones, for instance, would be inserted.

In what concerns autism spectrum disorder, this condition is by no means an exception to the importance of design. Taking this condition's idiosyncrasies into account, from a theoretic point of view, both Humanoid and Non-Humanoids SARs can be effective, although the design approaches tend to be linked to different goals.

Giving a humanoid form to socially assistive robots is a strategy usually used when striving for generalization, that is, for children to be able to replicate behavioural patterns learned in therapy sessions in different contexts, for instance, at school, when interacting with their peers. Overall, humanoid designs are considered particularly well-suited for imitation and emotion recognition activities. Nevertheless, there is no agreement on a design that is significantly better than others, although some general guidelines on robot designs have been proposed [30].

One of such guidelines infers that there is a delicate balance relative to how realistic a SAR's design should be. This principle is motivated by the fact that children diagnosed with autism tend to avoid social interaction with humans. Hence, whenever children with ASD see a robot with a human form, they often become withdrawn and avoid interactions. Robots in the forms of animals, cars and toys often do not trigger these same reactions, which can make them more engaging than robots with a humanoid form [30].

Furthermore, Robins et al. in [32] and [31], performed studies to evaluate the importance of the robot's appearance for children with autism, by evaluating the interactions between these children and several types of humanoid robots, which even included a human disguised a robot. This study revealed that robots that interact with children with autism should avoid the details and complexity of a human's appearance while still holding to the humanoid form.

On the one hand, in contrast to their humanoid counterpart, non-humanoid SARs are not affected by the problem of looking too much like a human, meaning children generally find it easier to interact with this kind of SAR design in a broader range of engaging activities since they will not feel intimidated by the robot's appearance. In this sense, it is thought that conceiving a simple-looking SAR can lead to an increase in the chance of a child with ASD to be engaged in the proposed activity, possibly yielding better results [30].

On the other hand, when using non-humanoid SARs, generalization (which is usually the main therapeutic goal) can be significantly reduced, as the amount of mimicking activities, such as hand gesture mimicking or head position mimicking, and "human-to-human" interactions are inadvertently reduced. This, however, does not necessarily constitute a problem. In reality, when the expected therapeutic outcome consists mainly in increasing a child's maximum concentration period (which is the case for this project), the use of non-humanoid robots may actually be the most efficient method due to the previously mentioned simplicity factor, as will be elaborated in Section 2.5.1

2.2.2. Adaptability

The second trait considered is adaptability, that is, the SAR's ability to adapt its behaviours in a personalized way according to each user. This trait has been stated in [25] to positively affect the robot acceptance, as well as increasing the robot use over time.

In fact, research has shown that challenging behaviors, such as distress or apathy in persons with cognitive impairments, occur more often in care settings that lack person-centered care. The main reason is that in person-centered care assistance is adapted to the individual's personality, along with his/her physical, psychological, and social needs [25].

After all, as stated in Chapter 1, there are several levels of ASD and the symptoms can vary substantially depending on the individual. Furthermore, regarding the likelihood of socially assistive robots being used in the long term, the fact that a patient's physical and mental needs might change over time supports the claim that the capacity of a SAR to adapt to each specific individual is just as important as its overall appearance since, in the end, even a perfectly shaped robot could not be considered particularly useful when it can be used only for a short time or limited number of therapy sessions for a limited number of people. Additionally, studies such as in [16] have shown that technologies which provide user-personalized experience tend to positively affect patient's attitude towards the technology at hand. This, in turn, influences the frequency of using the technology as well. Once again, this fact advocates that adaptability holds significant importance in promoting a positive and sustained use of technology, leading to higher levels of engagement during the assistive activities. Nevertheless, most robots in development today do not present a truly personalized experience, by considering each user's individual needs. Instead, their behaviour tends to be generalized across groups of people [25]. In this sense, it can be concluded that aside from choosing the right appearance, incorporating adaptability into any developed prototype not only holds paramount importance but can also be considered an innovative approach.

2.2.3. Simplicity

As described in [15], socially assistive robots must be easy to operate, taking into account that most clinicians are far from experts when it comes to coding. Hence, simplicity is fundamental for any interface presented to the user, which can either be the clinician or, depending on the type of therapy session in question, the patient. To put it briefly: the simpler the interface (e.g., a Graphical User

Interface or GUI) that is presented to the therapist, either to introduce commands or to access data registered by the robot, the better.

2.2.4. Responsiveness

Responsiveness can be defined as the therapist's ability to change the robot's behaviour during a therapy session. This trait is particularly relevant when dealing with people suffering from ASD, as these people will often not exhibit predictable patterns of interaction, may lose interest in the robot for periods of time or may persevere with interactions with the robot that are not therapeutically productive [15]. In such cases, it is fundamental for the therapist to be able to intervene with the highest degree of flexibility possible, with flexibility quantifying the range of possible actions the therapist might take, together with the speed at which any adjustments are implemented into the SAR's behavioural patterns.

2.2.5. Autonomy

The last characteristic that is crucial for therapeutic application of SARs is autonomy, or, in other words, the robot's ability to successfully perform independently from the therapist, (i.e. without continuously receiving new instructions on what its next action should be). A risk of a non-autonomous robot is that it prevents the therapist from paying enough attention to a patient or other therapeutic aspects that are not necessarily addressed by the SAR, during a therapy session, which may occur due to the high workload needed to keep the robot performing effectively.

Consequently, SARs must be programmed in such a way that all their programs or activities can be executed appropriately in a fully autonomous way, even when there is no feedback provided by the therapist (which is the most likely scenario since, as was just mentioned, the main focus of the therapist lies in analysing the patient's behaviour rather than actively correcting the robot's behaviour).

In light of the previous statements, a conflict inherent to the definitions of simplicity, autonomy and responsiveness may rise: on the one hand, ideally, the SAR would behave as an autonomous system requiring little to no attention from the therapists so that they can focus on the patient's response, whilst, on the other hand, the need for simplicity and responsiveness implies that therapists may need to alter the SAR's behaviour and the SAR should respond to these alterations as quickly as possible and in a comprehensive way (conflict between the concepts of autonomy and responsiveness). Moreover, the higher the degree of responsiveness, the higher the complexity of the subsequent system (conflict between the concepts of responsiveness and simplicity). Additionally, improving the autonomy of the system may increase its complexity (conflict between the concepts of autonomy and simplicity). Therefore, it can be concluded that there should be a trade-off between autonomy, simplicity and flexibility, depending on the considered therapeutic application.

2.3. SAR Examples

After considering all the previously mentioned theoretical aspects regarding the development of SARs, several concrete examples, regarding the use of SARs in therapeutic applications provided in literature will be discussed next

2.3.1. Mascot SAR: Charlie

The first example presented consists in Charlie, a socially assistive robot designed for interaction with children with ASD created by L. Boccanfuso et al. [5] and illustrated in Figure 2.2. Design-wise, according to the classes represented in Figure 2.1, this robot fits into the mascot class due to its cartoonish appearance. Additionally, a particularly simple appearance has been chosen for Charlie in order to engage the attention of children with ASD and to facilitate social interaction. This SAR was designed to conduct therapy sessions for children suffering from ASD, meaning it presents similar goals as pursued by this MSc project. The therapeutic method used, however, is slightly different from DMT, introduced in Chapter 1. Instead of attempting to get the child to move along with it, the robot focuses entirely on prompting the child's imitation skills, which, much alike DMT, have been linked to yield positive therapeutic results for children with ASD (in fact, the therapeutic measures performed by Charlie can be seen as simply a kind of movement therapy).

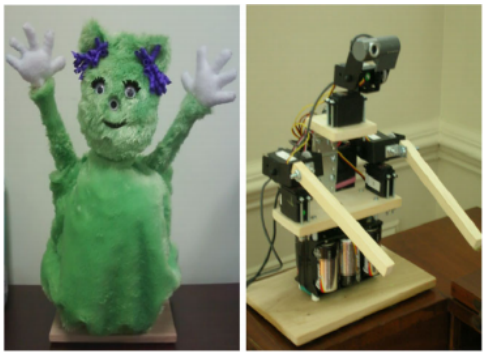


Figure 2.2: Charlie, a mascot-type SAR [5]

Figure 2.3: Paro, an animal-type SAR [21]

In order to recognize and properly interact with children, Charlie is equipped with a camera. Additionally, an image processing module consisting on face and palm classifiers are implemented to the robot. With a fully functional Image Processing Unit (IPU), Charlie is able to properly register the child's movements and, by using the its IPU, decide whether the child is imitating the robot correctly, and react to the child accordingly.

Charlie's decision making system (or control module), enabled two fairly simple game modes: a single player mode named "Imitate Me, Imitate You" and a multiplayer mode named "Pass the pose". In both modes, the child has the opportunity to interact passively with the robot, by imitating a move initiated by Charlie, or actively, where the robot waits for the child to make a move and subsequently imitates it.

Overall, taking into account the effective interactions between Charlie and the children who participated in L. Boccanfuso's study in [5], it is has been concluded that Charlie is a good example on how a simple-looking robot with fairly straightforward games can attract the attention of children with ASD, leading them to want to interact with it. Furthermore, [5] also addresses the importance of an accurate IPU. More specifically, the image processing regarding the hand and face movements are crucial for both the control module to work properly and for accurate registration of the data associated to the child's movements during a therapy session, which will be necessary for post-therapy analysis. The same fact holds for this thesis research project, where visual processing will hold paramount importance towards the end result.

Lastly, in light of the desired traits mentioned in Section 2.2, one particular drawback stands out: the fact that Charlie is not a truly adaptable robot. In fact, Charlie follows the exact same game pattern regardless of the child with whom it is currently interacting. After having stressed the importance of a personalized therapy session, it is believed that, when implementing a control module on the developed SAR, the robot should be able to adapt its behaviour to the child it interacts with during the therapy session.

2.3.2. Animal SAR: Paro the Seal Robot

Shibata et al. [35] used a baby harp seal as a model to design Paro, an animal SAR, which is illustrated in Figure 2.3. Paro has been designed in such a way that it is simultaneously appealing to users and can stimulate and sustain interaction. If classified according to the scale proposed by Daniel Ricks [30], regarding its appearance, Paro is an animal SAR.

Additionally, the robot is equipped with sensors acting as its four primary senses: sight (by using a light sensor), hearing (in the sense of determining the direction of the determination of sound source), balance, and tactile. The inclusion of these senses, whose readings influence the amplitude of the robots movements, guarantees Paro's responsiveness to the user, particularly in case of sudden stimulation. For instance, after a sudden loud sound, Paro pays attention and looks in the direction of the sound, illustrated by Paro turning its head towards the source of the loud sound.

Aside from the previously mentioned reactive behavior, Paro also exhibits a proactive behaviour using a system that is composed of two layers: a Behaviour-Planning layer and a Behaviour-Generation layer. The Behaviour-Planning layer consists on a state transitioning network based on the the robot's internal states, with each state being associated with a particular emotion that has been quantified

numerically. Furthermore, each state decays in time and is directly influenced by external interactions with a patient.

The behaviour-planning layer sends basic behavioural patterns (several poses and movements) to the behaviour-generation layer, depending on Paro's current state values. This second layer is subsequently responsible for generating control references for each actuator to perform the determined behaviour, with the control reference depending on magnitude of the internal states and their variation, creating an enormous number of emerging seemingly life-like behaviours. Naturally, this makes Paro's movements unpredictable, which increases the level of authenticity in the interactions. In addition, the behaviour generation layer is also responsible for adjusting the relative priority of reactive behaviours relative to proactive behaviours based on the magnitude of the internal states.

The interactions between the two mentioned layers already result in some degree of personalization, as different actions conducted by distinct patients will influence Paro's internal states (computed in the behaviour planning layer) in different ways, leading it to present distinct reactions. This, however, is not all. In order to keep traces of the previous interactions and to exhibit a coherent behaviour, Paro has a function of Reinforcement Learning (RL). RL will be described in detail in Chapter 3. It presents a positive value on preferred stimulation such as stroking and, in contrast, has negative value on undesired stimulation, such as beating. Consequently, in a gradual manner, Paro can be tuned to perform only preferred behaviours [21].

Unlike the previous example, the present socially assistive robot was not used in ASD therapy but instead in treating patients suffering from dementia. Even though the two conditions are fundamentally different, one trait of the SAR developed was particularly enticing: its ability to present a personalized behaviour for each of its users. In other words, the robot was significantly adaptive. It should additionally be noted that this was mainly achieved through the use of RL.

In light of the positive results obtained in [21], mainly due to Paro's adaptive behaviour, together with the high versatility and adaptability related to the use of RL algorithms, which could just as easily be applied for ASD therapy, it is considered that RL-based approach could be a positive feature for this MSc thesis project and should thus be considered.

2.3.3. Humanoid SAR: NAO

The final example considered in the current section consists in NAO, a humanoid robot illustrated in Figure 2.4. Design-wise, according to the scale presented in [30], NAO is considered to be a Mascot SAR, mostly due to its humanoid and cartoonish appearance. One possible drawback associated to this design choice consists on children tending to focus too much on the mechanical components of the robot, rather than focusing on the interactions with it.

In terms of dimensions and hardware, NAO is rather small, measuring only 0.57 m, and light and weighting 4.5 Kg. Its size relative to a child is showcased in Figure 2.5. Furthermore, much like the previous two examples, NAO relies on a set of sensors (including two camera, accelerometers and gyroscopes) to be able to perceive the environment around it, in particular, the children with whom it interacts. In terms of mobility, it is a fairly complex robot, possessing 11 degrees of freedom (DOF) for its lower limbs and another 14 DOF for its upper parts.

In their research, Shamsuddin et al. in [34] presented the development of a pilot experiment protocol where children with ASD were to be exposed to the humanoid robot NAO, where the aim would be triggering the children's interest in the robot during the therapy sessions. In the protocol presented, the robot is controlled in manual mode while a variety of modules (or games) are executed by NAO to entice reaction and interaction from the ASD children. These modules are defined before the therapy session using a built-in graphical user interface, with the actual therapy involving switching between the developed module set, in such a way that each child experiences each module only once. Additionally, as the robot is executing the activity modules, the initial response and behavior of the child with ASD is recorded. The recordings are then analyzed in a post-processing stage after the experiment.

According to Shamsuddin in [34], three people are needed during a therapy session: a manual operator, an occupational therapist and a psychologist. In the proposed experimental set-up, the manual operator works in a different room from the one where the therapy session takes place, and is consequently not visible to the children, the occupational therapist or the psychologist. The manual operator is thus responsible for three tasks: monitoring the video stream from the available external cameras (three were recommended) placed in the room where the therapy session is taking place; monitoring

the video stream from NAO's own vision system; and, lastly, manually controlling the robot.

Although the premise of testing NAO as a SAR is regarded as highly relevant for the scope of this MSc project, the experimental procedure is considered to fall short of the expectations due to several factors.

First and foremost, the need for three people to be actively involved in a therapeutic sessions was perceived as excessive, especially the manual operator. Taking into account the previously mentioned desired traits for a SAR, this usage of NAO is not considered to be sufficiently autonomous. Instead, the robot should be able to choose the most appropriate action by itself, following its decision making system, without any need to be continuously controlled by a human.

Secondly, the experimental setup proposed lacks personalization, as each one of the modules (or games) presented follows the exact same guidelines regardless of a child's preferences. The main differences occur in case the child becomes restless and uncooperative, which according to Shamsuddin, should result in terminating the current interaction and either moving on to the next game or finishing the experimental activity. Therefore, in light of the desired characteristics described in Section 2.2, the proposed usage of the robot lacks adaptability.

Although lacking adaptability and autonomy, the fact that NAO presents a GUI is seen in a positive light. In reality, by using this built-in functionality, therapists are given more control over NAO's behaviour when interacting with the child, which allows them, for instance, to set up personalized sequences of motions which could act as dance choreographies, and consequently increase the responsiveness. This stresses the importance of implementing a simple GUI meant to be used by the therapist as a way to increase responsiveness.

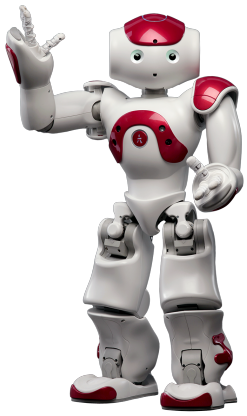


Figure 2.4: Humanoid Robot NAO [39]

Figure 2.5: NAO robot sitting next to a child [34]

2.4. Drones: Origin, Applications and Investment Boom

Over the last 20 years, Unmanned Aerial Vehicle (UAV) have radically increased their presence in the daily lives of people worldwide, particularly in developed countries. From their usually frowned upon wide range of military employments, to its generally acclaimed health related ones (e.g. recently being used to transport medical kits and Covid-19 tests between hospitals in the UK [1]), the number of applications of this seemingly new technology only seems to increase. This leads one to wonder: should therapeutic applications be considered as the next field to be revolutionized by relying on drone usage?

Although usually considered as state of the art due to the extensive media coverage it has been exposed to, strictly speaking, the concept of unmanned aerial vehicles is not properly new. In reality, the first drone, an aircraft named Hewitt-Sperry Automatic Airplane depicted in Figure 2.6, can be dated back to 1917, that is, during the first world war [8].

[1] Source: The Guardian, article by Aaron Walawalkar, October 2020 https://www.theguardian.com/technology/2020/oct/17/nhs-drones-deliver-coronavirus-kit-between-hospitals-essex

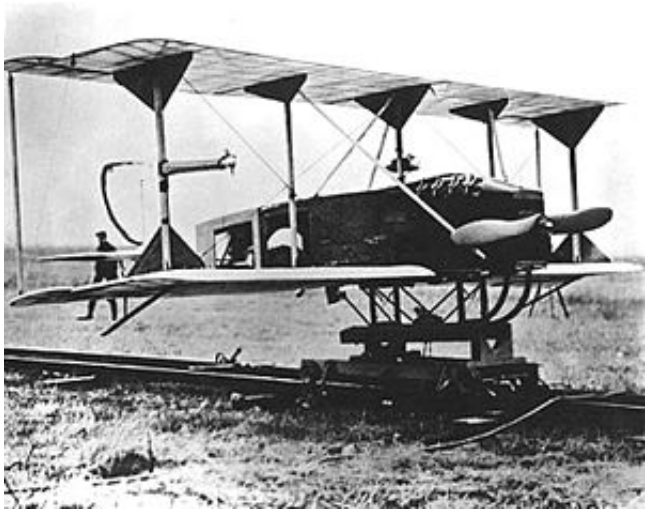


Figure 2.6: Hewitt-Sperry automatic airplane, 1917 [2]

Just as many other technologies, such as the global positioning system (GPS) or even the internet, initially, drones were used exclusively for military applications. Nevertheless, nowadays, the scenario is completely different, as drones are generally small, relatively inexpensive and easily available. The previous claim is further grounded by the graphs depicted in Figure 2.7.

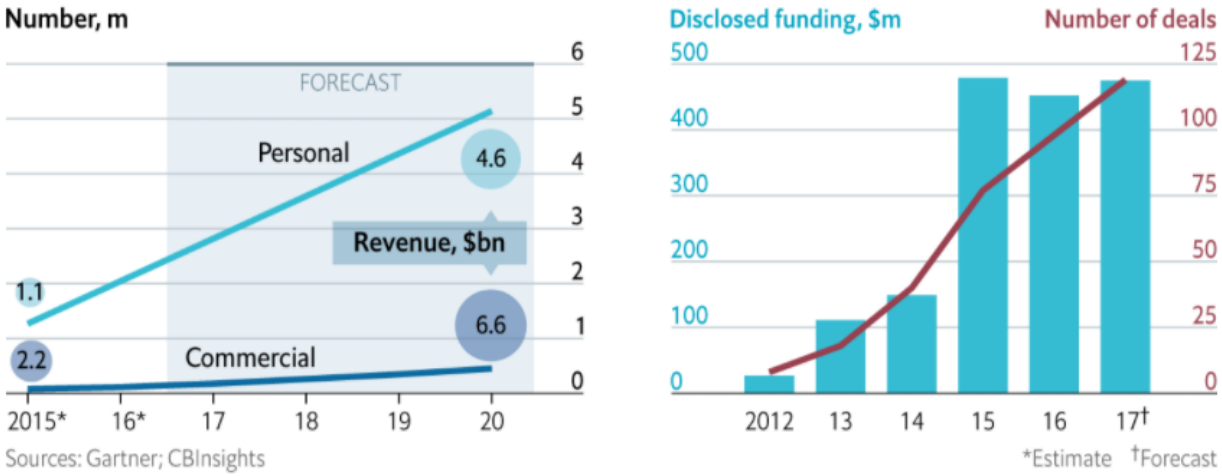


Figure 2.7: Civilian Drones Worldwide [3]

By carefully analysing the data presented in Figure 2.7, two main factors are made clear. Firstly, in light of the values presented in the graph on the left, there is an overall growth tendency for both personal and commercial drones (both considered owned by civilians). In fact, these are expected to reach a combined number of over 5 million units by the end of 2020. This is considered particularly impressive when taking into account that the same value, in 2015, fell short of the 2 million unit mark. Furthermore, the associated revenues are also expected to increase substantially, reaching 11 billion dollars in 2020, which, once again, presents a gigantic increase when compared to the 3.3 billion estimated in 2015.

Secondly and most importantly, by taking a look at the graph on the right, it can be clearly visualized that the increase in demand has been corresponded by a steep increase in disclosed funding. Considering the data presented, it can be seen that the value of disclosed funding went from little over USD 20 million in 2012 to USD 450 million in 2016, with estimates suggesting that this value will only

[2]Source: http://blogs.mentor.com/jvandomelen/blog/tag/hewitt-sperry-automatic-airplane/
[3]Source: The Economist Technology Quarterly June 10th 2017

increase in the near future. This increase has naturally resulted in a continuous increase in the number of deals, also depicted on the right graph in Figure 2.7.

Furthermore, in their report published in 2016: "Drones: reporting for work"[4], the American bank Goldman Sachs goes as far as to predict that a total of USD 100bn is likely to be spent on both military and civilian drones between 2016 and 2020. This report equally states that the commercial segment would be the fastest-growing, notably in construction (accounting for USD 11.2bn), agriculture (USD 5.9bn), insurance (USD 1.4bn) and infrastructure inspection (USD 1.1bn). It should additionally be noted that the latest data available as of March 2020 regarding yearly drone market investments in the US can be found in the Appendices, in Figure A.1.

2.5. Drones as Socially Assistive Robots

In light of the previously mentioned yearly increase in investment in drone related research, the number of drone applications has continually grown over the past few years. Knowing this while having mentioned the main characteristics required in a SAR in Section 2.2, it is now possible to address the question presented at the start of the current section, regarding drone effectiveness as SARs.

Firstly, it is crucial to mention that, to the best of my knowledge, drones have never been used as SARs, particularly in what concerns therapy related to children diagnosed with ASD. The considerations presented below will thus take into account which of the desired traits mentioned in Section 2.2 hold the most importance for the case in point.

Furthermore, the considerations presented below will mainly concern the use of a Parrot's Bebop 2 drone (depicted in Figure 2.8 and Figure 2.9), although they are considered equally valid for any drone with similar characteristics. In terms of specifications, the drone is relatively small (382mm [frontal length] x 328 mm [side width] x 89 mm [height]) and light, weighting around 500 grams.

Figure 2.8: Parrot Bebop 2: perspective view

Figure 2.9: Parrot Bebop 2: top view

2.5.1. Appearance

Bearing in mind the previous thoughts on the importance of appearance for SARs (Section 2.2.1), it is considered that, according to Figure 2.1, small-sized drones such as the Parrot Bebop 2 are inserted into the in the Non-Humanoid Mobile Robots category. Consequently, one of the major drawbacks associated with its therapeutic use consists on generalization of the behaviour learned in therapy being harder to achieve. In other words, it might be hard for any child involved in this sort of therapy to be able to replicate the social behaviour acquired in a therapy session.

In order to address this problem, one key aspect which must be considered consists in the main goal associated with the therapy sessions. If the main aim consists in the child being able to replicate specific interaction patterns learned with the help of a SAR (i.e. achieve behavioural generalization), then a Humanoid Robot would be a more indicate choice. However, in case the main therapeutic goal is

[4]Full report available at https://www.goldmansachs.com/insights/technology-driving-innovation/drones/

more comprehensive, (e.g. as increasing a patient's concentration period) then Non-Humanoid Mobile Robots take the lead over their humanoid counterparts. In fact, as stated in Section 2.2.1, a child with ASD will tend to be able to focus on a robot which does not resemble a human for longer periods of time, due to its not intimidating and enticing mechanical appearance.

In this sense, it can be concluded that, with the intent of increasing a child's attention span, a non-humanoid design, particularly a drone similar to the Bebop 2 model, would be an adequate choice.

2.5.2. Adaptability

After having stressed the benefits of developing an adaptive system as a means to attain a fully personalized experience, it is considered that a drone should be able to fully satisfy the required level of adaptability.

Nonetheless, it should be stated that the main factor responsible for guaranteeing adaptability is none other than the decision-making module. This module, or controller, acts as the drone's brain and will be discussed in detail in the following Chapter 3, together with several strategies to increase adaptability, such as RL or adaptive control.

In light of this, even if the controller mostly decides SAR adaptability, the fact that a small-sized drone is being used should not be totally undermined, as it carries some considerable advantages. Since the drone will be conceived to implement DMT, its inherent high mobility relative to other non-aerial robots should enable it to perform both simple choreographies or fairly more complex ones, depending on the user at hand. Naturally, this property presents itself as a good premise for the development of an adaptive controller.

2.5.3. Simplicity

In order to ensure that the program developed for interacting with the drone is simple to use for the therapist, it should be assumed that therapist working with the SAD is not comfortable with or even used to using any traditional programming interface. Consequently, a user friendly graphical user interface should be used instead of a fairly complicated one.

This trait, unlike the previous two, is not directly influenced by the use of a drone instead of the traditional SAR, since, in general, autonomous SARs should present an intuitive interface.

2.5.4. Autonomy

When developing a Socially Assistive Drone, guaranteeing a high level of autonomy stands as a priority. Ideally, once the drone is active and is given the start command by the therapist, it should be able to interact in a fully autonomous (and even intelligent) way with the patient, while registering all relevant data for posterior analysis.

Achieving the desired level of autonomy, once again, heavily relies on the control system implemented into the drone. The controller is responsible for determining the drone's optimal reaction to external stimulus, such as camera images, at all times. In this sense, the implementation approach is not conceptually different from that present in other autonomous robots.

Nonetheless, using a drone instead of any other form of SAR comes with a disadvantage. Although its small size enables high mobility, it also implies that the built-in battery must be small, which constrains therapy session length into a maximum of 20 minutes.

2.5.5. Responsiveness

In what concerns responsiveness, unlike autonomy, making sure that the therapist controls the drone in a flexible and recurrent way will not be one of the main priorities when developing the SAD controller. Instead, as was previously mentioned, the main goal is to ensure that the drone works in an autonomous, intelligent and personalized way, enabling therapists to focus on children's interactions with the drone, rather than on operating it.

With that said, therapists should be given a minimum level of flexibility to influence the drone's behaviour. Hence, even when considering highly autonomous drones, basic interactive options should be present in case a GUI is developed for the SAD.

2.6. Drone's Social Applications

Although, to the best of my knowledge, drones have never been used as socially assistive robots, the increasing interest and investment on these vehicles has lead to drones being employed in a wide range of recreational purposes. Some of these applications are considered to be getting significantly closer to the concept of SAR, which stands as another positive indicator for the potential drones hold in this field.

2.6.1. Drones for Live Streaming of Visuals for People with Limited Mobility

To illustrate the previous point, in 2016, E. Mangina et al. [20] developed software enabling live streaming of video footage captured by drones meant for people with limited mobility. According to the authors, the aim of the project consisted in using drones and virtual reality as surrogates to provide access to visual information to "differently-abled" people. Furthermore, in this case, drones are chosen as the ideal streaming platform because these are considered to offer a number of unique affordances to mobile technology research for community empowerment. This is essentially due to drones being relatively inexpensive, easy to operate and to fit with alternative interfaces for people of all abilities, and being readily available. Moreover, the fact that drones can carry a payload of light, inexpensive, and off-the-shelf sensors is also seen in a positive light as the data collected from these can be used to support a wide range of research efforts.

Regarding the reason why live streaming visuals is thought to be beneficial, in particular, for people with limited mobility, it is considered that, when taking advantage of the camera's high point of view, the viewers get a particularly good impression of most of the sites presented. In turn, this enables them to notice certain details which would not be easily perceived even during a "real experience" over the same landscape. According to Mangina et al. in [20], over longer periods of time, this experiment promises to "facilitate people with limited mobility to have access to live streaming of visuals and thus to experience the sense of empowerment and inclusion that live engagement in physical activity can inspire and support".

Overall, the developed system proposes to integrate Virtual Reality (VR) with a low-cost, unmanned, semi-autonomous quad-rotor. This quad-rotor, together with a VR headset, showcased in Figure 2.10, allows for first-person vision and manipulation using the Robot Operating System. Consequently, the system enables the user to move the quad rotor remotely using natural head movements, which are then tracked by the VR headset and translated into six degrees of freedom commands. These commands are subsequently sent to the drone, giving a person with limited mobility full control over the drone's actions.

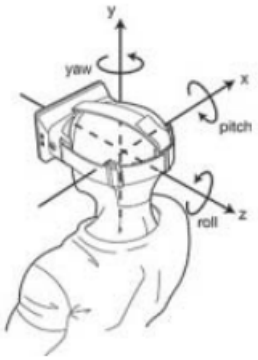


Figure 2.10: Oculus Rift Sensor Setup [20]

Considering the project description so far, it must be stated that although this research does strive to provide the participant with a greater sense of empowerment, belonging, and achievement, it cannot be said that this use of drones helps the patient when it comes to rehabilitation. For this reason, the current employment of drones is not compatible with the definition of SAR. Nonetheless, this research is regarded as an important example showcasing the possibilities associated with drone use, particularly cheap and easily available quad rotors, for counteracting specific personal impairments, such as limited mobility.

2.7. Final remarks

In conclusion, during the previous chapter, the concept of socially assisted robots was thoroughly analysed, along with several examples that have been developed so far. Additionally, the main required characteristics for a successful therapeutic implementation were examined. Lastly, drones were introduced as potential SARs, taking into account all the aforementioned key traits required: appearance, adaptability, simplicity, responsiveness and autonomy.

Considering all this, it can now be safely stated that drones hold great potential as SARs. Nonetheless, as was hinted throughout the section, a successful implementation of drones in a therapeutic context heavily relies on the SAD's decision-making module, responsible for ensuring that traits such as adaptability and autonomy. For this reason, developing the optimal decision-making module is critical. Thus, the following chapter will address the drone's decision making module. In particular, it will explore the available types of controller available, considering their main advantages advantages and disadvantages given the project scope.

3

State of the Art Control Methods

The current chapter focuses on the decision-making module of the drone, which will henceforth be referred to as the drone's control system, or simply the drone's controller. In this sense, this chapter addresses the technical and mathematical properties related to several kinds of controllers by introducing control theory concepts such as adaptive control, fuzzy logic control, reinforcement learning and artificial neural networks. Furthermore, examples of drone controllers using these methods, often combining two or more, are presented and discussed.

Nonetheless, before addressing the previous concepts, a brief overview of control system theory is provided.

3.1. Introduction to control systems

According to K. Ogata in [26], a control system can be defined as a procedure to control, that is, continually manage a set of specified variables within a given process. The controlled set of variables, (which are designated as controlled variables) quantifies the property which the control system attempts to control. Furthermore, the control input is defined as the quantity or condition that is introduced by the controller so as to affect the value of the controlled variable which is often the output of the system. Hence, control means measuring the value of the controlled variable and applying a control input to the system to correct or limit deviation of the measured value from a desired value.

In this sense, a controller can be interpreted as an input-output mapping which receives the error between the current measured state variable, x, and the current state reference x_{ref}, $e = x_{\text{ref}} - x$, its time derivative $\frac{de}{dt} = \dot{e}$ or the time integral, $\int_{t_0}^{t} e(t)\,dt$. A typical feedback control loop is represented below, in Figure 3.1.

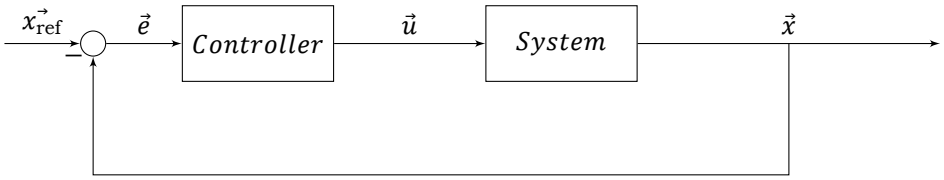


Figure 3.1: Block diagram of a standard controller in an ideal system

It should be noted that, in the previous diagram, the state x is represented as an n-dimensional vector, $\vec{x}$, with each element representing one of the n states that the controller aims to track. Similarly to $\vec{x}$, $\vec{e}$ is represented as an n-dimensional vector, whereas $\dot{e}$ and $\int e\,dt$ are usually computed within the controller block, using numerical methods, only when needed, and are thus not illustrated.

Concerning the control input, represented as $\vec{u}$ in Figure 3.1, this value represents the signal sent from the controller to the actuator which directly influences future states within the system block.

When modelling a controller for real hardware implementation, two additional factors need to be taken into consideration. Firstly, the fact that the actuators are not perfect must be taken into account. This means that, in a real scenario, the control input is not immediately implemented to the actuator. Instead, there's a transition period between a previous actuator position and the latest control input, which can vary significantly depending on the actuator considered. For this reason, an actuator model which implements the distinction between the desired actuator deflection and control input, $\vec{u}_{\text{ref}}$, and the real actuator deflection or variation, $\vec{u}_{\text{a}}$, is necessary.

The second point to take into account when considering implementation on real hardware is the sensors present in the system or process for which the controller is developed. In an ideal scenario, where the implemented sensors are perfect, the measured state would equal the real state. However, this is not a realistic approach, especially when it comes to drones, which, in exchange for a high availability at relatively low costs, tend to employ poor sensors with high variances in measurements. Thus, in reality, there is a discrepancy between the real state $\vec{x}$ and a measured state $\vec{x_m}$, which highly relies on the quality of the sensors implemented. Consequently, a sensor block should be placed into the diagram.

It should additionally be noted that sensor blocks are often accompanied by estimator blocks. These are responsible for correcting inaccurate state measurements taken by the sensors by taking into account their variance and bias. An example of an estimator often found in literature is the Kalman filter, thoroughly described by R. Mehra in [24]. Although it should be stressed that estimators play an important role in developing a control system by influencing the control loop, throughout the following section these are considered to be associated with the sensors block, as estimation techniques are not considered to belong in the project scope, presented in Chapter 1.

In light of the previous two factors, a second block diagram, presented in Figure 3.2, depicts a more accurate representation of the control loop, now including both the actuator and the sensor blocks.

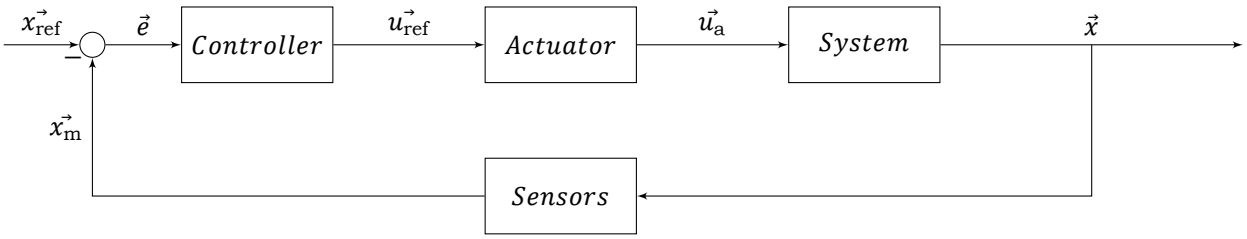


Figure 3.2: Block diagram of a standard controller in a real system

Out of the four blocks represented above, the research conducted in the following sections focuses entirely on the controller block, as the remaining three are tied to the hardware present on the robot (in this case, the drone) for which the controller is designed and are thus immutable. State-of-the-art control approaches include using artificial neural networks, reinforcement learning based agents or fuzzy logic systems. The following section addresses the latter.

3.2. Fuzzy Logic Controllers

The first control method described consists in Fuzzy Logic Controllers (FLCs), whose base theory was first introduced in 1965 by Zadeh [11]. The main idea explored thought the use of fuzzy logic is that of uncertainty, represented through the use of fuzzy sets. A fuzzy set A on a domain X is defined by the membership function $\mu_A(x)$, which is a mapping from the Universe X into the unit interval:

$$\mu_A(x) : X \longrightarrow [0, 1] \tag{3.1}$$

The value of the membership function for a certain object x is referred to as the membership degree and it represents to which degree object x belongs to fuzzy set A. Fuzzy sets are thus an extremely useful tool to quantify linguistic terms such as "hot", "cold" or "warm", for instance, when referring to a variable like temperature, or "far", "near" and "adequate" when referring to target distance. Variables

like the previously mentioned two, which are associated with fuzzy sets, are referred to as linguistic variables [42].

As for the membership functions, although these can follow any given analytical expression, some particular shapes are frequently found in literature, with the most common ones being triangular membership functions, trapezoidal membership functions, bell-shaped (including Gaussian) membership functions and, lastly, singleton membership functions. These four examples are illustrated below, in Figure 3.3.

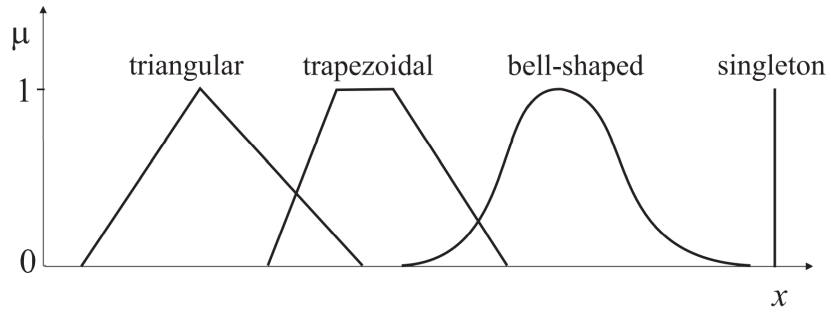


Figure 3.3: Common membership function shapes [28]

By finding a mathematical representation of linguistic terms, abstract concepts become quantifiable. This fact, in turn, enables defining linguistic rules, represented as R, according to an implication pattern following "if-then" rules described below, for a generic rule R_i:

$$R_i : \textbf{If } x \textit{ is } A_i \textbf{ then } y \textit{ is } B_i \qquad i = 1, 2, \dots K \tag{3.2}$$

where x is the input or antecedent linguistic variable, A_i is the antecedent linguistic term represented by a fuzzy set, y is referred to as the consequent (output) linguistic variable and B_i is the consequent linguistic term represented by a fuzzy set. In a similar way, the first part of the implication "if x is A_i" is referred to as the antecedent, whereas the latter half, "then y is B_i" is known as the consequent. It should additionally be considered that in the previous expression aside from A_i and B_i both the input and output x and y should also be interpreted as fuzzy sets. This comes from the fact that a real number (also referred to as crisp value) is a special case of a fuzzy set (singleton set) [42].

The rule format presented in (3.2) can be further developed by taking into account several inputs through the use of connectives such as "and" or "or", resulting in:

$$R_i : \textbf{If } x_1 \textit{ is } A_{i1} \textbf{ and } x_2 \textit{ is } A_{i2} \dots \textbf{ and } x_p \textit{ is } A_{ip} \textbf{ then } y \textit{ is } B_i \qquad i = 1, 2, \dots K \tag{3.3}$$

$$R_i : \textbf{If } x_1 \textit{ is } A_{i1} \textbf{ or } x_2 \textit{ is } A_{i2} \dots \textbf{ or } x_p \textit{ is } A_{ip} \textbf{ then } y \textit{ is } B_i \qquad i = 1, 2, \dots K \tag{3.4}$$

where these connectives are implemented through the use of t-norms (e.g. the minimum function) for the connective "and" and s-conorms (e.g. the maximum function) regarding the "or" connective. It should additionally be taken into account that even though (3.3) and (3.4) exclusively use either "and" or "or", these two connectors can be combined in the same rule, R_i.

The ability to mathematically define if-then rules is the essence of any FLC, a type of Knowledge Based Controller (KBC) which can be reduced to a set or rules identical to the ones presented above, in (3.3) and (3.4), along with their inherent antecedent and consequent fuzzy sets and membership functions.

Regarding the output computational process associated with an FLC, it consists on using the mathematical concept of implication. Although there are several implication algorithms available, such as the Lukasiewicz implication, the Kleene-Diene implication or the Larson (product) implication, the one that is most typically used in literature is the Mamdani (or minimum) implication, where the membership degree of each rule R, defined as μ_R, follows:

$$\mu_R(x,y) = I(\mu_A(x), \mu_B(y)) \tag{3.5}$$

with the Mamdani (also referred to as minimum) implication being defined as:

$$I(\mu_A(x), \mu_B(y)) = min(\mu_A(x), \mu_B(y)) \tag{3.6}$$

The previous Equations (3.5) and (3.6) form the basis for the max-min inference, also known as the Mamdani inference, an algorithm to compute the output fuzzy set B' given an input fuzzy set A' and a Rule Base $\mathcal{R}$ [17].

3.2.1. Mamdani Controller

The Mamdani controller is the main example of fuzzy controller that is explored in the current research, together with Takagi-Sugeno controller. This controller is illustrated in the form of a block diagram in Figure 3.4,

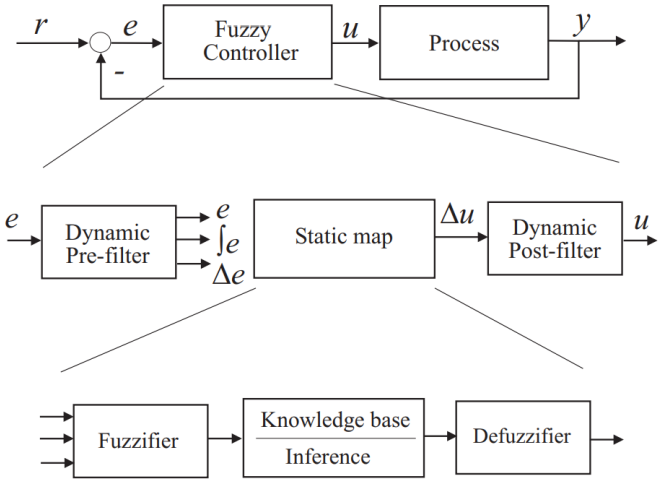


Figure 3.4: Fuzzy controller implementation in the control loop [28]

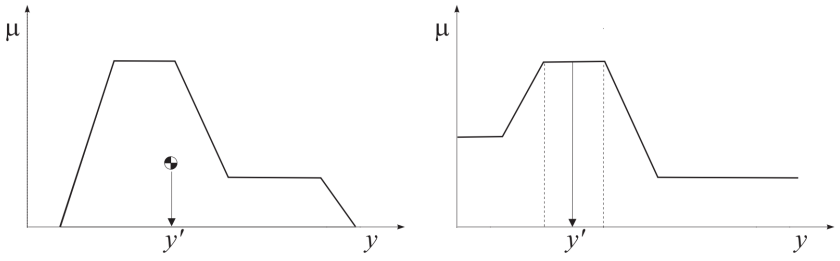
As it was previously mentioned, by using the aforementioned concept of implications (in particular, the max-min inference) on fixed set of rules defined *a priori*, it is possible to attain an output fuzzy set B' from an input fuzzy set A'. Nonetheless, when the controller is implemented into the control loop, its inputs, represented in Figure 3.4 as the error, its temporal derivative and its time integral (obtained through a dynamic pre-filter) are crisp values instead of fuzzy sets. To address this issue, a preliminary Fuzzification procedure must be met, where the crisp values are turned into fuzzy sets. This is usually done by interpreting the crisp value as a singleton fuzzy set. Nevertheless, other methods, such as using a triangular shaped membership function centered around the crisp value with a specified width value, can also be used.

After the inputs have been successfully fuzzified on the fuzzifier block, a knowledge based inference (e.g. the previously discussed mamdani inference) is used to generate an output fuzzy set B' from the existing rule set R and the fuzzified input A'. Once B' is obtained, it needs to be transformed back into a crisp value since the control input u must also be a crisp value. In order to achieve this, a process of defuzzification, represented by the defuzzifier block is implemented. Similarly to fuzzification, there are several defuzzification processes, with the ones more frequently used being Mean of Maxima (MOM) and Center of Gravity (COG), represented in Figure 3.5 to obtain the defuzzified value and control input y' [17].

Following the defuzzication process, the controller output is then sent to the system's actuators.

3.2.2. Takagi-Sugeno Controller

The Takagi-Sugeno (TS) controller (often referred to as Takag-Sugeno-Kang (TSK)) was first introduced in 1985 by T. Takagi and M. Sugeno in [37] and it works very similarly to the Mandami controller. The



(a) Center of gravity. (b) Mean of maxima.

Figure 3.5: Illustration COG (a) and MOM (b) defuzzification methods [28]

main difference consists on how the fuzzy rule consequents are defined. Instead of using a linguistic term to describe the output, y, it is defined by an analytical expression based on the inputs, $\mathbf{f}_i(x)$. Consequently, a generic rule R_i is defined as:

$$R_i : \textbf{If } x \ is \ A_i \textbf{ then } y_i = \mathbf{f}_i(x) \qquad i = 1, 2, \dots K \tag{3.7}$$

where $\mathbf{f}_i(x)$ is typically either a constant (Zero Order TS controller) or a linear combination of the inputs (First Order TS controller).

The output of a controller with K rules is computed by taking the average of all outputs y weighted by each rule's membership degrees, according to:

$$y = \frac{\sum_{i=1}^{K} \mu_{R_i} y_i}{\sum_{i=1}^{K} \mu_{R_i}} \tag{3.8}$$

The previous consequent format enables a more precise definition of the output computational process, $\mathbf{f}_i(x)$, when compared to the standard mamdani controller. This property can be particularly useful when the desired expressions for $\mathbf{f}_i(x)$ are known for each linguistic term expressed in the antecedent. In other words, it is convenient when the controller's desired behaviour for each input interval is well known, resulting in a more accurate control over the process. Furthermore, the TS controller, unlike the more general mamdani controller, does not need to go through a defuzzification process, as the output y is already a crisp value instead of a fuzzy set. Overall, this controller format is less intuitive in comparison with its generic mamdani variant but, on the other hand, it enables users to implement on the controller more detailed knowledge regarding the process, though $\mathbf{f}_i(x)$.

3.2.3. Advantages of FLC

Through the implementation of fuzzy rules in a fuzzy logic controller, expert *apriori* knowledge can be implemented into the system. This property is responsible for the humanlike behaviour of any KBC, since humans controllers, unlike the standard automatic controllers such as PIDs, do not require mathematical models nor exact trajectories to control any given process. Instead, controllers tend to follow a fixed set of rules learned either through hands-on experience or through extensive study on the matter at hand. Consequently, this ability to resemble human decision making often makes KBCs (including FLCs) the most suitable control method when replacing a human operator. Nonetheless, the fact that FLCs tend function similarly to a human operator implies that, in certain scenarios, their performance may not be as exact as that exhibited by a controller with access to a perfect mathematical model of the system.

Furthermore, these controllers are more intuitive than the traditional control approach, whose performance highly relies on tuning gain values that usually do not hold any meaning with respect to the process at hand.

In addition to being simpler and more humanlike, FLCs do not require the process' mathematical model, which is often difficult to obtain, depending on the process' inherent complexity. Alternatively, by defining simple knowledge based rules dictating how the controller should act in each situation described by user-defined linguistic terms, a satisfactory performance can be easily obtained. Nonetheless, it should also be stated some a priori knowledge is in fact needed, meaning that FLCs might not be the most suitable approach for an unknown process.

Lastly, the fact that these controllers are not associated with a mathematical model makes them more effective when dealing with time variant non-linear processes. In such cases, by definition, the processes' model changes in time, preventing a classical PID controller from working. This property is considered to be one of the most valuable properties associated with the use of an FLC.

The previous considerations over the advantages and disadvantages of using FLCs are summarized below, in Table 3.1.

Advantages of FLCs	Disadvantages of FLCs
• Easy to implement knowledge into the system • Exhibits humanlike behaviour • Intuitive fuzzy rule approach • No mathematical model needed • Efficiently dealing with nonlinearities	• A priori system knowledge needed for fuzzy rules • Not adequate for high precision tasks

Table 3.1: Listing of main advantages and disadvantages associated with Fuzzy Logic Control

3.3. Adaptive Controllers

Not all processes can be sufficiently well-controlled using controllers with a fixed set of parameters, where most FLCs and PID controllers are included. In order to address this issue, a different type of approach was developed. In an adaptive controller, its inherent parameters are tuned online, that is, while interacting with the process, in order maintain the required performance despite unforeseen changes in the external conditions [1]. As Astrom mentions in his paper "Theory and Application of Adaptive Control" [2], the concept of adaptive control is not particularly new, as it was first introduced in the 1950s. Since then, several optimization algorithms for adaptive controllers were developed, in other words, algorithms devised to continuously change the controller parameters (e.g. the proportional, derivative or integral gains in case of a PID controller). In particular, gain scheduling, model reference adaptive systems and, lastly, self-tuning controllers are thoroughly described in the previously mentioned article and are be briefly addressed next.

3.3.1. Gain Scheduling

The first of the three adaptive control schemes presented by Astrom in [2] is the gain scheduling approach. This method relied on the premise that in certain cases it is possible to find auxiliary variables which correlate well with the changes in the process' dynamics, i.e. with variations in the process's parameters. After identifying such variables, it becomes possible to reduce the effects caused by parameter variations by changing the parameters of the controller (which is referred to as regulator in [2]) as functions of the auxiliary variables. The name gain scheduling refers to the fact that this adaptive approach was originally used to change the gain of the controller implemented. Furthermore, this adaptive method is depicted in the form of a block diagram in Figure 3.6

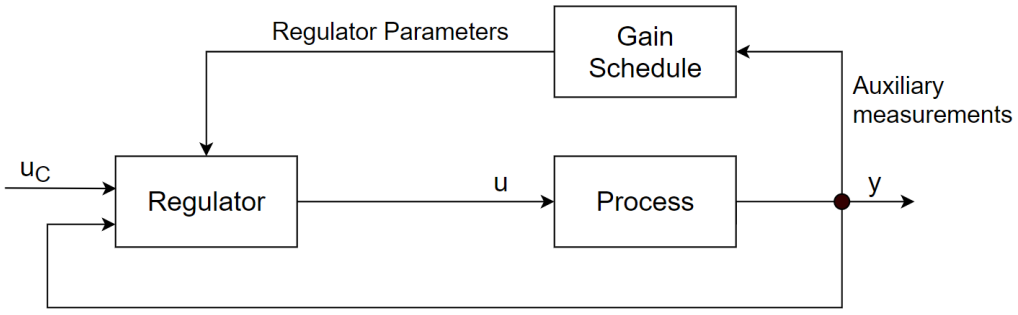


Figure 3.6: Block diagram illustrating the Gain Schedule adaptive method

In the previous block diagram, u_c represents the system reference, u the control input, and y the process output.

Although gain scheduling presents an effective adaptive method, its main drawback consists in finding suitable scheduling variables. As mentioned by Astrom in [2], this search process is not always simple and it heavily relies on the physics of the considered system.

3.3.2. Model Reference Adaptive Systems

The second adaptive approach described by Astrom in [2] consists in Model Reference Adaptive System (MRAS) and is shown in Figure 3.7. In this case, the diagram involves developing a process model which computes how the process output should ideally respond to a given input u. This ideal output is represented as y_M. Additionally, it should be noticed that the reference model is part of the control system. Consequently, the controller can be thought of as consisting of two loops: the inner and outer loops. Concerning the inner loop, it is an ordinary control loop composed of the process and the controller. The parameters of the controller are adjusted by the outer loop, in such a way that the error e, between the model output, y_m, and the process output, y, becomes small. The outer loop is thus also a regulator loop.

Regarding the controller's parameter adjustment mechanism, which is responsible for maintaining system stability while bringing the error, e, to 0, the so-called 'MIT-rule', presented in (3.9) was used.

$$\frac{d\theta}{dt} = ce\nabla_\theta e \tag{3.9}$$

In (3.9), the components of the vector θ denote the adjustable controller parameters, that is, the values which are continuously adjusted by the outer control loop. Furthermore, the components of the vector $\nabla_\theta e$ are the sensitivity derivatives (i.e. gradient) of the error with respect to the set adjustable parameters θ. The sensitivity derivatives can be generated as outputs of a linear system driven by process inputs and outputs. Lastly, c is a parameter which determines the adaptation rate.

The main assumption behind this algorithm consist in supposing that the parameters θ change much slower than the remaining system variables. Additionally, it is expected that in order to minimize the squared error, e^2, the controller's parameters should change in the direction of negative gradient of e^2. Thus, (3.9) can be rewritten as:

$$\theta(t) = -c\int_{t_0}^{t} e(s)\nabla_\theta e(s)ds \tag{3.10}$$

The previous equation (3.10) shows that the adjustment mechanism can be thought of as composed of three parts: a linear filter for computing the sensitivity derivatives from process inputs and outputs ($\nabla_\theta e$), a multiplier and an integrator.

The MRAS adaptive control system is known for delivering good performance as long as the adaptation rate, c, is small, as stated in [2]. The allowable magnitude for c differs in each case, depending on the magnitude of the reference signal. Therefore, it is not possible to give fixed limits which guarantee system stability, meaning that the MIT-rule can thus give an unstable closed-loop system. It should additionally be noted that just as is the case with gain tuning, in order to use MRAS effectively, a significant amount of information on the system must be known. This knowledge is represented in the block diagram in Figure 3.7 within the model block, which must output fairly accurate results in order for the adaptive algorithm to work.

3.3.3. Self-tuning Controller

A third adaptive control approach is mentioned by Astrom in [2]. This method is known as the self-tuning regulator (or Self-Tuning Controller, STC) and is represented as the block diagram in Figure 3.8. Just as is the case with the MRAS, this approach can be thought of as being composed of two loops: an inner loop and an outer loop. The inner loop consists of the process and an ordinary linear feedback controller. The parameters of the controller are adjusted by the outer loop, which is composed of a recursive parameter estimator and a design regulator. Additionally, in order to obtain realistic estimates, stochastic perturbation signals are often used within the parameter estimator block. This function, however, is not shown in Figure 3.8 in order to maintain simplicity.

Furthermore, it should be noted that the block named ”regulator design” represents an on-line solution to a design problem for a system with known parameters that have been previously estimated by the parameter estimator block. Naturally, in order to design the contents of this block, the desired values of the controller's parameters need to be stipulated (either through mathematical models or functions, or through the use of fuzzy logic) for each set of process parameters that are estimated. This implies that, similarly to the gain scheduling and MRAS methods, the STC approach also assumes that the process that is being controlled is fairly well-known.

Concerning the recursive parameter estimator block, many different estimation schemes have been used, such as stochastic approximation, least squares, extended and generalized least squares, instrumental variables, extended Kalman filtering, etc.

Due to not imposing any restrictions on the methods used in both the parameter estimation phase and the controller design phase, this method is regarded as a highly flexible method. Additionally, it is considered fairly easy to understand, allows designing a personalized approach for each case and can be smoothly implemented into a microprocessor.

Regarding the relation between the STC and the MRAS, it is considered that these two approaches focus on developing different kinds of control systems for different kinds of processes. Whereas the MRAS was originally obtained by considering a deterministic servo problem, the STC considers a stochastic regulation problem. Nonetheless, by comparing the schemes depicted in Figure 3.7 and Figure 3.8 it is clear that these two approaches are closely related. Both systems present two feedback loops, with the inner loop being a standard feedback loop containing simply the process and the controller. Additionally, in both methods, the controller presents adjustable parameters which are set by the outer loop, with the adjustments being based on the feedback from the process' inputs and outputs. Overall, both require significant knowledge regarding the process at hand. However, even if the two approaches are similar, the STC approach is considered to be more adequate due to its higher degree of flexibility (no fixed algorithm) and ability to incorporate noise into the process, which is considered to be more realistic.

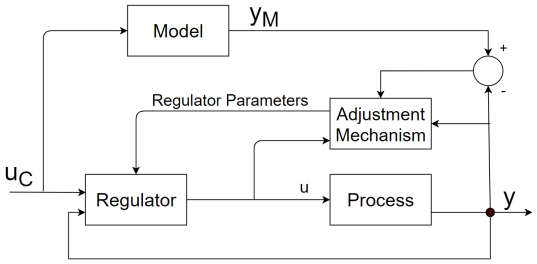


Figure 3.7: Block Diagram illustrating a Model Reference Adaptive System

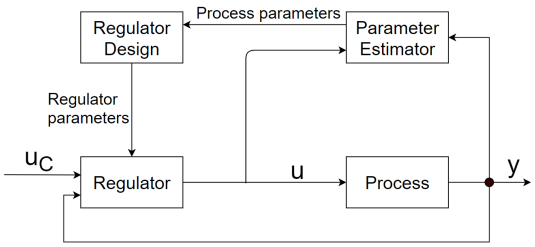


Figure 3.8: Block Diagram Illustrating the Self-Tuning Controller approach [2]

3.3.4. Advantages of Adaptive Control

By using an adaptive control approach, the scope of scenarios where controller performance is adequate increases. This increase is justified by the fact that the controller continuously adapts its parameters in order to match the changes in the process. Consequently, if there are significant disparities in the process parameters in time caused by varying external conditions (i.e. according to the traditional control theory, the poles of the system move), the controller remains tuned. Naturally, this leads to maintaining an adequate performance in a broader range of situations.

3.4. Fuzzy Logic Adaptive Controllers

So far, the main benefits of both fuzzy logic control and adaptive control have been discussed. In order to integrate the advantages of using an adaptive controller to those associated with an FLC, the two controller concepts can be combined. Merging these two concepts results in a KBC centered on fuzzy rules and fuzzy logic (thus an FLC) that is additionally able to update (or adapt) its fuzzy rule-base while being used (thus an adaptive controller). The resulting controller is referred to as a Fuzzy Logic Adaptive Controller (FLAC). This controller incorporates an adaptive controller's increased performance

while retaining FLC's simplicity and ease in implementing knowledge into the system, without the need for an accurate process model.

In the previous Section 3.3, a PID with adjustable proportional, derivative and integral gains was mentioned as an example of an adaptive controller. This classical type of controller is mainly characterized by the previous three gains, meaning that identifying the parameters which are continuously adjusted is rather simple. However, when dealing with an FLC, characterized by a set of linguistic terms, membership functions and rules, selecting the variables (or values) which are continually tuned is not as straightforward. This problem is particularly significant when considering FLCs with a high number of inputs and rules, which implies the existence of hundreds of membership functions. In such cases, due to each membership function being defined by at least two parameters (Figure 3.3), the number of adjustable variables could reach several thousands, even when considering a static number of rules. In case the number of rules is allowed to vary, along with the number and type of connectives (such as the "and" or "or"), the number of adjustable parameters only further increases.

Taking into account the number of adjustable parameters in the controller, two distinct scenarios regarding FLACs are described. Firstly, a simpler case is considered, where the fuzzy rule set is small, resulting in a small number of linguistic terms and input variables, together with a reduced and fixed set of rules, $\mathcal{R}$. Additionally, it is assumed that there is a mathematical model of the process available and the process itself is sufficiently well-known. In this situation, the number of adjustable parameters, consisting on membership function parameters (e.g. standard deviation in case of a gaussian membership function), is significantly reduced. Due to this reduced number of adjustable parameters, an adaptive strategy similar to the ones presented in Section 3.3 is expected to yield good results. This can happen due to two reasons: either there is a good correlation between the set of adjustable parameters and some of the system variables, which would justify using a gain scheduling approach; or, as an alternative, in case the process is sufficiently well-known and the mathematical model is accurate enough, both the MRAS and STC are expected to yield good results. Overall, in this scenario, it is considered that the previous approach would result in a fairly effective controller, although potentially lacking in case high precision outputs are required.

The second situation considered for FLAC implementation concerns a more complex system (or process) where no mathematical model is available. Additionally, in this scenario, a controller with higher precision is required. As was previously mentioned, the adaptive approaches explored so far require *a priori* knowledge regarding the process at hand, either through the existence of a model or through the existence of known correlations between process variables and adjustable parameters. Consequently, when little is known about the system, implementing adaptive approaches such as gain scheduling, MRAS or STC stops being a viable approach. Moreover, the requirement of increased precision in a fuzzy controller leads to an increased complexity when developing the fuzzy rule-sets. This, in turn, results in an increasing set of adjustable controller parameters, subsequently increasing the overall controller complexity.

In order to implement an FLAC in the second scenario, new types of adaptive algorithms must be considered. Ideally, these alternative adaptive state of the art techniques would be able to interactively learn from the external unknown environment, as well as be capable of accurately tuning a wider range of adjustable controller parameters. Therefore, in the following sections, three algorithms that allow interactively learning from either the external environment or available data are introduced. These are Artificial Neural Networks (ANNs), Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL).

3.5. Reinforcement Learning

As defined by Sutton and Barto in "Reinforcement Learning: An Introduction" [36], Reinforcement learning consists in "learning what to do - how to map situations to actions - so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards."

3.5.1. Markov Decision Processes

Markov decision processes (MDPs) are regarded as a straightforward framing of the problem of learning from interactions to achieve a pre-defined goal. In an MDP the learner (i.e. decision maker) is called the agent; the external conditions with which the agent interacts, including everything outside the agent, are referred to as the environment. Due to the nature of these two concepts, they interact continually, with the agent selecting an action, a, and the environment responding to these actions by presenting a new situation to the agent, represented as a state, s. It should be pointed out that, in order to be consistent with the notation used by most of the scientific community, in an RL context, the state will be represented as s, whereas in a more comprehensive control related scenario (as was done so far), the state vector is represented as $\vec{x}$. Additionally, the environment is responsible for rewarding the agent. A reward, $r \in \mathbb{R}$, is a numerical value that the agent seeks to maximize over time through its choice of actions. These interactions are summarized in Figure 3.9.

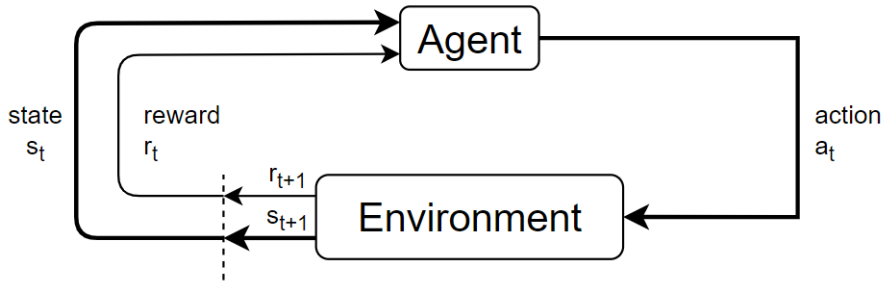


Figure 3.9: Agent-environment interaction in an MDP

In a Markov Decision Process, the interactions between the agent and the environment rely on discrete time steps, t. At each time step, the agent receives a representation of the environment's state, $s_t \in S$, and takes that information into account to select an action, $a_t \in A(s)$. Subsequently, in the following time step, the agent receives a numerical reward $r_{t+1} \in \mathbb{R}$ and reaches an updated state s_{t+1}.

By definition, in a finite MDP, the sets of states, actions, and rewards (S, A, and R) all have a finite number of elements. Consequently, the random variables r_t and s_t have well defined discrete probability distributions dependent only on the preceding state and action. This property is summarized in (3.11) and (3.12) [36].

$$p(s', r|s, a) := \Pr\{s_t = s', r_t = r|s_{t-1} = s, a_{t-1} = a\} \tag{3.11}$$

$$\sum_{s' \in S} \sum_{r \in R} p(s', r|s, a) = 1, \text{ for all } \; s \in S, a \in A(s) \tag{3.12}$$

where $s' \in S$ represents any given state which can be achieved from the current state $s \in S$.

In an MDP, the probabilities given by p completely characterize the environment's dynamics. That is, the probability of each possible value for s_t and r_t depends only on the immediately preceding state and action, s_{t-1} and a_{t-1}, and, given them, not at all on earlier states and actions. This is best viewed as a restriction not on the decision process, but on the state. The state must include information about all aspects of the past agent-environment interaction that make a difference for the future. If it does, then the state is said to have the Markov property [36].

3.5.2. Discounted Return

In RL, the agent's goal is to maximize the total amount of reward it receives. This means maximizing not immediate reward, but the expected value of the cumulative reward in the long run. In the simplest case, this return is the sum of the rewards:

$$G_t = r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_T \tag{3.13}$$

with T denoting the final time step.

This approach is particularly useful when considering applications in which there is a natural notion of final time step, that is, when the agent-environment interaction breaks naturally into subsequences, named episodes. Consequently, each episode ends in a terminal state, followed by a reset to a standard starting state or to a sample from a standard distribution of starting states. Tasks with episodes of this kind are called episodic tasks. [36].

In contrast, it is also possible that the agent-environment interaction does not break naturally into identifiable episodes, but goes on continually without limit. For instance, this would be the natural way to formulate an on-going process-control task. These types of interactions are called continuing tasks. In such cases, the return formulation (3.13) is problematic for continuing tasks because the final time step would be $T = \infty$, meaning the return could itself be infinite. To address this issue, the additional concept of discounting is often used. In this approach, the agent tries to select actions so that the sum of the discounted rewards it receives over the future is maximized. Consequently, the agent chooses its action to maximize the expected discounted return [36]:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{3.14}$$

where γ represents the discount rate, with $0 \leq \gamma \leq 1$.

The discount rate determines the present value of future rewards: a reward received k steps in the future is only worth γ^{k-1} times what it would be worth if it were received immediately. It should be noted that, if $\gamma < 1$, the sum described in (3.14) presents a finite value as long as the reward sequence r_K is bounded. The magnitude of the discount rate is thus related to how farsighted the agent is. If $\gamma = 0$, the agent is only concerned with maximizing the immediate rewards, whereas when γ approaches 1, the return objective takes future rewards into account more strongly.

It should additionally be noted that from (3.14), it follows that returns in successive time steps are related to each other according to:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = r_{t+1} + \gamma G_{t+1} \tag{3.15}$$

3.5.3. Policy and Value Function

A vast majority of reinforcement learning algorithms involves estimating value functions, that is, functions of states (or of state-action pairs) responsible for estimating ”how good” it is for the agent to be in a given state. Consequently, the value function relies on the expected return. Due to the expected rewards depending on the agent’s actions, value functions are defined with respect to particular ways of acting, called policies.

As defined by Sutton and Barto in [36], formally, a policy is a mapping from states to probabilities of selecting each possible action. If the agent is following policy π at time t, then $\pi(a|s)$ is the probability that $a_t = a$ if $s_t = s$.

The value function of a state s under a policy π, denoted $V_\pi(s)$, is the expected return when starting in s and following π thereafter. For MDPs, this results in:

$$V_\pi(s) = \mathbb{E}_\pi[G_t|s_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right], \text{ for all } s \in S \tag{3.16}$$

where $\mathbb{E}_\pi$ represents the expected value of a random variable (in this case the expected reward) given that the agent follows a policy π, and t is any time step.

Similarly, the value of taking action a in state s under a policy π, denoted as $Q_\pi(s, a)$, is defined as the expected return starting from s, taking the action a, and thereafter following policy π:

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t|s_t = s, a_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a\right] \tag{3.17}$$

where Q_π is called the action-value function for policy π.

The value functions V_π and Q_π can be numerically evaluated from experience. Although there are several methods of conducting the estimations (Monte Carlo methods, temporal difference methods,

dynamic programming) the following Section 3.5.5 focuses on the temporal difference approach, particularly the Q-learning algorithm.

For finite MDPs, it is possible to define the concept of optimal policy, denoted as π^*, referring to a policy that is better or equal than all other policies, π w.r.t. the value function. A policy π is considered better or equal than another policy π' if and only if $V_\pi \geq V_{\pi'}$ for all $s \in S$. Although an optimal policy is usually referred to as a single policy, it should be noted that there can be more than one. All the optimal policies share the same realized value for the state-value function (denoted as optimal state-value) and the same optimal realized values for the action-value function. The optimal state-value and action value mappings (or functions in a continuous scenario) are respectively defined as:

$$V^*(s) = \max_\pi V_\pi(s) \quad \text{for all} \quad s \in S \tag{3.18}$$

$$Q^*(s,a) = \max_\pi Q_\pi(s,a) \quad \text{for all} \quad s \in S, \quad a \in A \tag{3.19}$$

Regarding the optimal action-value function, it represents the expected return for taking an action a in state s and thereafter following the optimal policy. Taking in to account 3.16 and 3.17 and Bellman's principle of optimality [36], it can be written in terms of V^* as:

$$Q^*(s,a) = \mathbb{E}[r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a] \tag{3.20}$$

In order to obtain V^*, the Bellman equation must be solved. This equation expresses a relationship between the value of a state and the values of its successor states. In fact, it states that the value of the start state must equal the (discounted) value of the expected next state, plus the reward expected along the way. For the value function of a generic policy π, V_π, this equation is written as:

$$V_\pi = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)\Big[r + \gamma V_\pi(s')\Big], \quad \text{for all} \quad s \in S \tag{3.21}$$

When concerning the optimal value function, V^*, the Bellman equation is referred to as the Bellman optimality equation. Intuitively, this equation expresses the fact that the value of a state under an optimal policy must equal the expected return for the best action from that state:

$$V^*(s) = \max_a \sum_{s',r} p(s', r|s, a)\Big[r + \gamma V^*(s')\Big] \tag{3.22}$$

where, in this case, the double sum presented in (3.21) was condensed into a single sum for simplicity.

Similarly, the Bellman optimality equation for Q^* is defined as:

$$Q^*(s,a) = \sum_{s',r} p(s', r|s, a)\Big[r + \gamma \max_{a'} Q^*(s', a')\Big] \tag{3.23}$$

As proven by Sutton and Barto in [36], for finite MDPs, the Bellman optimality equation for V^* (3.22) has a unique solution independent of the policy. Consequently, if the dynamics p of the environment are known, then, in principle, one can solve this system of equations for V^*. Furthermore, a similar set of equations can be solved in order to obtain Q^*. Once V^* is known, the optimal policy can be found: for each state s, there will be one or more actions at which the maximum is obtained in the Bellman optimality equation. Any policy that assigns nonzero probability only to these actions is an optimal policy.

Explicitly solving the Bellman optimality equation provides one route to finding an optimal policy, and thus to solving the reinforcement learning problem. However, this solution is rarely useful. This is due to this method relying on three assumptions that are rarely true: firstly, it assumes that we accurately know the dynamics of the environment; secondly it takes for granted that we have enough computational resources to complete the computation of the solution (which, depending on the dimension of the state-space, may take years on state-of-the-art computers); and, lastly, it assumes the Markov property.

In order to address this issue, several methods of approximately finding a solution to the Bellman optimality equation were created, with one of the most widely used being Q-learning.

3.5.4. Temporal Difference Learning

Temporal Difference (TD) methods focus on using experience to solve the prediction problem, in other words, to estimate the state value function of a policy π, $V_\pi(s)$. The available estimate for V_π is denoted as $\hat{V}$. The simplest TD method consists in updating the value function for an arbitrary state at time t, $\hat{V}(s_t)$, according to:

$$\hat{V}(s_t) \leftarrow \hat{V}(s_t) + \alpha \left[r_{t+1} + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t) \right] \tag{3.24}$$

where parameter α is called learning rate, with $\alpha \in]0, 1]$

From (3.24) it follows that each $\hat{V}(s)$ is immediately updated at time step $t+1$, using the observed reward r_{t+1} and the estimate $\hat{V}(s_{t+1})$, which are weighted by the learning rate α and the discount rate γ. The TD method described by (3.24) can also be referred to as TD(0) method. Furthermore, because TD(0) bases its update in part on an existing estimate, it is called a bootstrapping method.

Additionally, the amount added to a previously existing state-value $\hat{V}(s_t)$ during updates is referred to as the TD error, being defined as:

$$\Delta_t = r_{t+1} + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t) \tag{3.25}$$

In other words, Δ_t is the error in $\hat{V}(S)$, available at time $t+1$.

TD learning methods are seen as a combination of dynamic programming methods and Monte Carlo methods, merging the main advantages of both. Consequently, TD methods are particularly prominent in RL. The main advantages associated with using TD algorithms consist, firstly, in TD methods being able to update estimates based in part on other learned estimates, without waiting for a final outcome. This enables them to be implemented in an online, fully incremental fashion, with online meaning that the agent learns by interacting with the process, as opposed to an offline learning approaches where data is collected in advance. Secondly, like Monte Carlo methods, TD methods can learn directly from raw experience without a model of the environment's dynamics [36].

Regarding the soundness of the bootstraping concept, in other words, whether it can be mathematically assured that learning one guess from the next, without waiting for an actual outcome, guarantees convergence to the correct answer, the answer yes. In fact, is has been proven that for any fixed policy π, TD(0) converges to V_π, in the mean for a constant step-size parameter (in this case, α) if it is sufficiently small, and with probability 1 if the step-size parameter decreases according to the usual stochastic approximation conditions expressed in 3.26 [36].

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \quad \text{and} \quad \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty \tag{3.26}$$

where the first condition is required to guarantee that the steps are large enough to eventually overcome any initial conditions or random fluctuations. The second condition guarantees that eventually the steps become small enough to assure convergence.

Considering a case where an offline learning approach is taken, meaning there is only a finite amount of experience available, a typical method consists in presenting the experience repeatedly until the TD prediction algorithm converges upon an answer. This kind of approach, denoted as batch updating, is compatible with the TD(0). In fact, under batch updating, TD(0) converges deterministically to a single answer independent if the step-size parameter, α, as long as α is chosen to be sufficiently small. However it should be noted that the answer found by the TD(0) is not necessarily optimal, it all depends on the amount and quality of the data presented [36].

3.5.5. Q-learning

Considering the use of TD prediction methods for the control problem at hand, several algorithms have been developed, including SARSA and Q-learning. The current section addresses the latter.

In Q-learning (initially described by Watkins in [40]), the goal is to learn an action-value function rather than a state-value function, according to:

$$\hat{Q}(s_t, a_t) \leftarrow \hat{Q}(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a \hat{Q}(s_{t+1}, a) - \hat{Q}(s_t, a_t) \right] \tag{3.27}$$

It should be noticed that, in Q-learning, the learned action-value function, $\hat{Q}(s_t, a_t)$, directly approximates Q^*, the optimal action-value function, regardless of the policy being followed. This trait is responsible for Q-learning being considered an off-policy method. In contrast, methods where $Q_\pi(s, a)$ is estimated for the current behavior policy π and for all states s and actions a are designated as on-policy methods (e.g. SARSA).

The fact that the Q^* can be obtained regardless of the policy being followed by the agent dramatically simplifies the analysis of the algorithm, enabling early convergence proofs, as suggested by Sutton in [36]. Furthermore, it should be taken into consideration that although the state-value function of the policy the agent follows is not particularly relevant, this policy still has an effect in that it determines which state-action pairs are visited and updated. In order for the algorithm to converge, it is necessary for all state action pairs to be continually updated. The previous condition is considered a minimal requirement in the sense that any method guaranteed to find optimal behavior in the general case must require it.

Thus, one of the challenges inherent to most reinforcement learning algorithms, including Q-learning, is the trade-off between exploration and exploitation. The dilemma is that neither exploration nor exploitation can be pursued exclusively without failing at the task. This means that the agent must try a variety of actions and explore the state-action space (exploration), while progressively favor those that appear to be best in order to ensure a fast convergence (exploitation). In short, if a policy explores too much, it may take a significant amount of time and computational power to converge and if it explores too little, the algorithm may not find the optimal answer.

Nonetheless, assuming that the agent sufficiently explores the environment and a variant of the usual stochastic approximation conditions on the sequence of step-size parameters (expressed in (3.26)), $\hat{Q}$ has been shown to converge with probability 1 to Q^*. The Q-learning algorithm is summarized below in a procedural form.

Algorithm 1: Q-learning for estimating $\pi \approx \pi^*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$;
Initialize $\hat{Q}(s, a)$ for all $s \in S$, $a \in A$, arbitrarily except that $\hat{Q}$(terminal,.) = 0;
for each episode do
  Initialize s_0;
  for each step of episode do
    Choose a from A using policy derived from Q (e.g. ϵ-greedy)
    Take action a, observe r, s_{t+1}
    $\hat{Q}(s, a) \longleftarrow \hat{Q}(s, a) + \alpha[r + \gamma \max_a \hat{Q}(s_{t+1}, a) - \hat{Q}(s, a)]$
    $s \longleftarrow s_{t+1}$
    if s is terminal then
      break;
    end
  end
end

3.6. Artificial Neural Networks

Artificial neural networks are widely used for nonlinear function (or system) approximation. As defined by A. Barto and R. Sutton in [36], an ANN is a network of interconnected units that have some of the properties of neurons, the main components of nervous systems. Although having a long story, it was only recently, with the recent advances in training deeply layered ANNs (deep learning) that this concept became responsible for some of the most impressive abilities of Machine Learning (ML) systems, in particular RL systems, discussed in Section 3.5.

In Figure 3.10, a generic feedforward ANN is represented. In this case, feedforward means that there are no paths within the network by which a unit output can influence its own input. In contrast, if an ANN has at least one loop in its connections, it is considered a recurrent rather than a feedforward ANN. It can be seen that the represented ANN, exhibits four inputs, represented as $x_i, i \in \{1, 2, 3, 4\}$

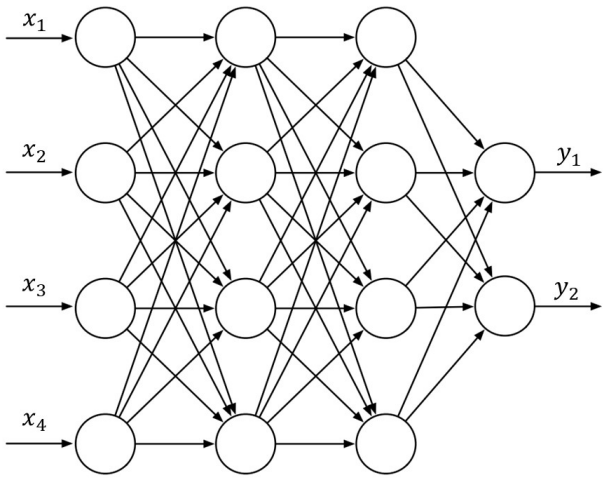


Figure 3.10: Feedforward ANN with four input units, two output units and two hidden layers [36]

and two outputs, $y_j, j \in \{1, 2\}$. Furthermore, the illustrated network presents four layers: an input layer, an output layer and two hidden layers, with a hidden layer being defined as any layer between the input and the output layers. A real value weight, w, is associated with each link, represented in the figure as an arrow, with its value corresponding to the efficacy of a synaptic connection in a real neural network.

The unit circles represented in Figure 3.10 are semi-linear units, meaning that they compute the weighted sum of their respective inputs and subsequently apply a nonlinear function to the resulting value. This function is named the activation function, and produces the unit's output, or activation. Furthermore, it is also common in ANN for each unit to add a specific bias, b, to the result of the weighted sum before computing its respective output through the activation function. Similarly to weights, biases also represent adjustable real values. A variety of different activation functions can be used, however, these are typically either S-shaped functions, such as the sigmoid function, defined as $f(x) = (1 + e^{-x})^{-1}$, or the rectified linear unit (ReLU) function, defined as $f(x) = \max\{0, x\}$. These functions are graphically illustrated in Figure 3.11 and Figure 3.12, respectively.

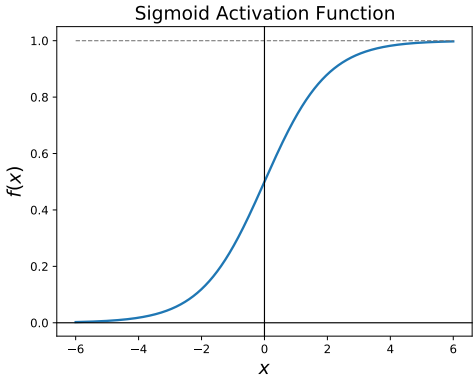


Figure 3.11: Graphical representation of the Sigmoid activation function

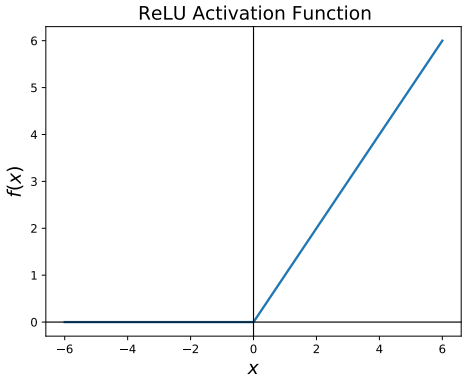


Figure 3.12: Graphical representation of the ReLU activation function

As referred by Barto in [36], the activation of each output unit of a feedforward ANN is thus ultimately a nonlinear function of the activation patterns over the network's input units. Furthermore, it should be noticed that an ANN with no hidden layers can only represent a very small fraction of the possible input-output functions. Nonetheless, in [9], Cybenko proved that an ANN with a single hidden layer containing a large enough finite number of sigmoid units can approximate any continuous function on a compact region of the network's input space to any degree of accuracy. The previous property is also valid for other nonlinear activation functions that satisfy mild conditions, with non-linearity being

essential.

The so-called ”universal approximation” property of ANNs with one hidden layer does not guarantee that restricting ANN to one hidden layer is the optimal approach. In fact, as described in [4], both experience and theory show that approximating the complex functions needed for many artificial intelligence tasks is made easier (and indeed may require) abstractions that are hierarchical compositions of many layers of lower-level abstractions, that is, abstractions produced by deep architectures such as ANNs with many hidden layers. The successive layers of a deep ANN can be interpreted as a way to compute increasingly abstract representations of the network's “raw” input, with each unit providing a feature contributing to a hierarchical representation of the overall input-output function of the network [36].

3.6.1. Artificial Neural Network Training: Gradient Descent Approaches

So far, ANNs have been presented as having great potential as tools to model nonlinear functions and systems. Nonetheless, the issue of how an ANN is tuned to accurately represent a given system has not been addressed, yet. Over the years, several algorithms have been developed to tune ANN, with most of them relying on gradient descent approaches (similar to the MIT-rule in Section 3.3.2). The general traits of this approach are now briefly introduced.

Firstly, it should be considered that training the hidden layers of an ANN can be interpreted as a way to automatically create features appropriate for a given problem ”so that hierarchical representations can be produced without relying exclusively on hand-crafted features” [36]. In order to achieve this degree of customization, the ANN's parameters must be tuned to match given scenarios. These parameters consist in the weights and biases of the ANN.

In gradient-descent methods, the weight vector is a column vector with a fixed number of real valued components, d, $\vec{w} = (w_1, w_2, ..., w_d)^\top$, and the function that the network attempts to approximate is differentiable with respect to $\vec{w}$ for all n states (or inputs) $\vec{x} = (x_1, x_2, ..., x_n)^\top$. In order to introduce the basic premise of deep reinforcement learning, described in detail in Section 3.7, let us assume that the function outputed by the ANN is the value function of a certain policy π, $V_\pi(\vec{x}, \vec{w})$. Through the successive iterations of the gradient-descent method, $\vec{w}$ is updated at each of a series of discrete time steps, $t = 0, 1, 2, ...$ in case of online learning. Alternatively, in case of offline learning, the iterative process makes use of available training data.

From an offline learning perspective, firstly, a cost function represented by J is defined, that is, the function that the ANN attempts to minimize. Although there are several alternatives, the (mean) squared error (MSE) is often used:

$$J(\vec{w}) = \frac{1}{2}(V_{\pi_i}(\vec{x}) - \hat{V}_{\pi_i}(\vec{x}, \vec{w}))^2 \tag{3.28}$$

where $\hat{V}_{\pi_i}(\vec{x}, \vec{w})$ is the estimate provided by the ANN for a piece of training data labeled with the target value, $V_{\pi_i}(\vec{x})$.

In order to minimize the error, the gradient of the cost function J with regard to the weight vector $\vec{w}$ is computed for a certain training input. By definition, $\nabla J(\vec{w})$ is a vector containing the partial derivatives of J with respect to each weight w_i.

$$\nabla J(\vec{w}) = \left(\frac{\partial J(\vec{w})}{\partial w_1}, \frac{\partial J(\vec{w})}{\partial w_2}, ..., \frac{\partial J(\vec{w})}{\partial w_d} \right)^\top \tag{3.29}$$

After computing $\nabla J(\vec{w})$ by using the chain rule, the weights can be adjusted following (3.30). Naturally, computing all the partial derivatives that constitute takes into account the ANN's architecture (e.g. number of hidden layers, number of neurons and connections between them) and properties (e.g. activation funtions).

$$w_{k+1} = w_k - \alpha_n \nabla J(w_k) \tag{3.30}$$

where α is the algorithm's learning rate or step size and $k \in \mathbb{N}$ represents the iteration number.

The weight update condition expressed in (3.30) is responsible for the method being referenced as ”gradient descent”, since the overall increment in w_k is proportional to the negative gradient of the example's squared error. This is, by definition, the direction in which the error falls most rapidly.

Furthermore, it should be noted that in some variations of the gradient descent method may include additional parameters used to decrease computational effort until convergence or increase the algorithm's stability. Such is the case of using momentum techniques (i.e. increasing step size in case the weights continually change in the same direction) or RMSProp (i.e. decreasing the step size over time, particularly when gradients are large).

When referring to offline learning, where a significant amount of data is available, a variant of the gradient descent method is typically used. In order to take into account the content of the whole available dataset when computing the gradient $\nabla J(\vec{w})$, instead of computing it sample-by-sample (using the previous equations) an estimate of the gradient of the whole data set is computed. In such cases, gradient descent methods are called "stochastic" as the gradient computation is done using a subset of data which has been selected stochastically from the total available data. By only selecting a random amount of data (called a batch) when computing an estimate for the gradient of the whole data set, computational effort is reduced and convergence is accelerated.

In this last scenario, the batch error is defined as:

$$J = \frac{1}{K}\sum_{i=0}^{K}\frac{1}{2}(V_{\pi_i}(\vec{x}) - \hat{V}_{\pi_i}(\vec{x},\vec{w}))^2 \tag{3.31}$$

with K being the total amount of samples in a batch (also referred to as batch size).

Furthermore, for the stochastic gradient descent method, the batch gradient, used to update the weights as expressed in (3.30) is given by the average of the K gradients computed for each sample, according to (3.29). This results in:

$$\nabla J(\vec{w}) = \frac{1}{K}\sum_{i=0}^{K}\nabla J(\vec{w})|_{V_{\pi_i}} \tag{3.32}$$

Lastly, it should be noted that although biases were not introduced in the previous equations, these may be present in the ANN used. Nonetheless, in such cases, the procedure for tuning the ANN is identical. In such occasions, the adjustable parameters vector, $\vec{w}$, should contain both the weights and the biases inherent to the ANN architecture. Consequently this leads the gradients taking into account the partial derivatives with respect to the biases in addition to the partial derivatives with respect to the weights.

3.7. Deep Reinforcement Learning

In the previous Sections 3.5 and 3.6, the concepts of ANN and RL were introduced. Both of these methods enable interactively learning from a given process. Concerning ANNs, these can model most processes, even those that are highly non-linear, provided that a large enough data set containing labeled data is provided. Due to the need for large amounts of input-output labelled pairs as data, ANNs are considered an example of supervised learning. RL-based algorithms learn by continuously interacting with the external environment until a course of action (policy) which maximizes the discounted expected reward is found.

In light of this, it can be seen that the scopes of RL and ANNs address different scopes of a given problem. For instance, putting it in terms of a standard RL problem, an ANN is responsible for modelling the environment, while RL is used to determine the best course of action in it.

The concepts of ANN and RL can be combined, resulting in deep reinforcement learning. Overall, DRL is a more comprehensive concept that includes the implementation of ANNs as function estimators (e.g. using an ANN to estimate the value function of a given policy, V_π). These estimators are subsequently combined with a given RL algorithm similar to the ones mentioned in Section 3.5.5 to find the optimal policy, π.

In addition to being a more comprehensive approach, DRL also presents itself as an answer to some of the problems inherent to classical RL strategies, in particular, the need for discretizing the state-action space. In Section 3.5, MDPs were introduced and associated with TD methods (e.g. Q-learning). Consequently, the TD methods explored so far assume a finite and discretized state action

space in order to operate. However, in most control problems, this assumption is not realistic, especially when considering complex control problems, where states and actions are continuous. Therefore, in such cases, implementing algorithms as traditional Q-learning would require discretizing the problem, yielding enormous state-actions tables (also referred to as Q-tables) where the Q-values for each individual discretized state-action pair are stored. Naturally, the high dimensionality (associated with required precision or simply with a large number of states and actions) would make the process computationally too expensive to fully explore the state action-space, meaning it would be impossible to find the optimal policy, π^*.

Deep reinforcement learning addresses this issue by proposing RL algorithms which can be applied to continuous state-action spaces, as is the case of the actor-critic method, which is explored in Section 3.7.1.

3.7.1. Actor-Critic Methods

Actor-critic RL methods have been introduced to deal with continuous state and continuous action spaces. In actor-critic methods, the value function and the policy are separated. The value function is represented by a unit called the critic, whereas the policy is represented by the actor unit. Usually both of these units consist in ANNs. The role of the critic is to predict the outcome of a particular control action in a given state of the process [36].

The control policy is represented separately from the critic and is adapted by comparing the reward actually received to the one predicted by the critic. A block diagram of a classical actor-critic scheme, which was first introduced by Barto et al. in [3], is depicted in Figure 3.13.

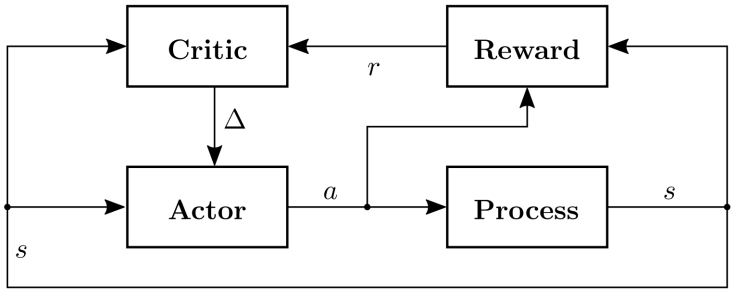


Figure 3.13: Actor-Critic learning scheme [28]

In addition to the external process, Figure 3.13 showcases three blocks which are the essence of the actor-critic approach: the reward block, the critic and the actor.

Regarding the reward block, it is responsible for providing the reward function to the system for each discrete time-step t, r_t. The reward function is also called the external reinforcement.

Concerning the critic module, its task is to predict the expected future reinforcement r that the agent will receive for being in the current state and following the current control policy, π. In other words, the critic is trained to predict the future value function $V_\pi(s_t)$ for the current state s_t. This prediction is then used to obtain a more informative signal, called the internal reinforcement, which is involved in the adaptation of the critic and the actor, by changing in the current policy. Due to changes in the behavioural policy followed by the agent, and the subsequent computation of the value function for each policy attempted, this method is an on-policy algorithm [36].

Denoting as $\hat{V}_\pi(s_t)$ the predictor of $V_\pi(s_t)$, the adaptation law for the critic can be simply derived by writing $V_\pi(s_t)$ as follows:

$$V_\pi(s_t) = \sum_{n=0}^{\infty} \gamma^n r_{t+n+1} = r_{t+1} + \gamma V(s_{t+1}) \tag{3.33}$$

In order to train the critic, the prediction error at discrete time step t, denoted as $\Delta_t = V_\pi(s_t) - \hat{V}_\pi(s_t)$ is computed. Although the true value function is unknown, it can be approximated by replacing $V_\pi(s_t)$ in (3.33) by its predictor, $\hat{V}_\pi(s_t)$, yielding the following estimate of the prediction error:

$$\Delta_t = V_\pi(s_t) - \hat{V}_\pi(s_t) = r_{t+1} + \gamma \hat{V}_\pi(s_{t+1}) - \hat{V}_\pi(s_t) \tag{3.34}$$

Since Δ_t is computed using two consecutive estimate values, the actor-critic method is considered a temporal difference method. It should additionally noted that both predictions $\hat{V}_\pi(s_t)$ and $\hat{V}_\pi(s_{t+1})$ are known at time $t+1$, since $\hat{V}_\pi(s_{t+1})$ is a prediction obtained for the current process state.

After being computed, the temporal difference error, Δ_t, serves as the previously mentioned internal reinforcement signal and is used to adapt the critic. Since the critic consists in an ANN, its prediction $\hat{V}_\pi(s_{t+1})$ can also be represented as $\hat{V}_\pi(s_{t+1}, \theta_t)$, where θ_t represents a vector of adjustable parameters (i.e. weights and biases). To update θ_t, a gradient-descent learning rule (Section 3.6.1) is applied, resulting in:

$$\theta_{t+1} = \theta_t + \alpha_c \Delta_t \frac{\partial \hat{V}_\pi(s_t, \theta_t)}{\partial \theta_t} \tag{3.35}$$

where α_c is the critic's learning rate.

Moreover, it is worth noticing that the expression obtained in (3.35) is obtained from (3.30) when considering a squared error (or cost) of $J = \frac{1}{2}\Delta_t^2$. Directly applying the chain rule on the error's derivative with respect to a given adjustable parameter ($\frac{\partial J}{\partial \theta_t} = \frac{\partial J}{\partial \Delta_t}\frac{\partial \Delta_t}{\hat{V}_\pi}\frac{\partial \hat{V}_\pi}{\partial \theta_t}$) results in the gradient vector being multiplied not only by the learning rate but also by the prediction error, Δ_t. Consequently, the bigger the prediction error, the bigger the correction applied to the adjustable parameters vector.

Regarding the actor module, it represents the policy π being followed at a given time. This block updates itself in accordance with the internal reinforcement provided by the critic. Consequently, while the critic is trained to predict the future system's performance (the value function), the actor (i.e., the policy) can be adapted in order to establish an optimal mapping between the system states and the control actions. This adaptation is done using the same Δ_t, as defined in (3.34).

The adaptation process can be summarized as follows: given a certain state s_t, the control action a_t^π is calculated using the current policy $\pi(s_t)$. This action is not directly applied to the process, but it is instead modified to obtain the effective action, a_t, by adding exploration $\tilde{a}_t$ to it. The exploration $\tilde{a}_t$ can, for instance, be a random value from a normal distribution with a standard deviation of σ, represented as $N(0, \sigma)$. After the modified action a_t is sent to the process block, the temporal difference Δ_t is computed, according to 3.34. If the actual performance is better than the predicted one, the actor is adapted toward the modified control action a_t.

Once again, assuming the actor block as an ANN, the action taken by the agent at each time step is given by:

$$a_t = a_t^\pi + \tilde{a}_t = \hat{\pi}(s_t, \varphi_t) + \tilde{a}_t \tag{3.36}$$

where φ is the vector of adjustable parameters of the ANN within the actor block. This vector is updated according to:

$$\varphi_{t+1} = \varphi_t + \alpha_a \Delta_t \tilde{a}_t \frac{\partial \hat{\pi}(s_t, \varphi_t)}{\partial \varphi_t} \tag{3.37}$$

where α_a is the actor's learning rate.

It should be noted that this time, in order to speed up convergence, the gradient is multiplied by the random action deviation $\tilde{a}_t$. Consequently, the bigger the deviation, the larger the correction applied to the adjustable parameters.

3.8. Examples of State of the Art Controllers

The previous sections within Chapter 3 focused on discussing the theory behind state of the art controllers. Henceforth, a more pragmatic approach is taken, by analysing several examples of controllers based on the theory explored so far.

3.8.1. Position Control of a Quadcopter Using Adaptive TS Controller

In "Position control of a quadcopter drone using evolutionary algorithms-based self-tuning for first-order Takagi-Sugeno-Kang fuzzy logic autopilots", Edward Yazid et al. [41] develop a controller for

position control of a quadcopter drone. The quadcopter drone selected is a Multiple Input Multiple Output (MIMO) system with highly non-linear rigid body dynamics and severe cross-couplings. The controller developed is an adaptive TSK controller, as introduced in Section 3.2.2. Regarding the adaptive mechanism three evolutionary algorithms are tested and compared, namely, Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC). The most relevant experimental results obtained in this research are showcased in Appendix B. In particular, the control system's block diagram of the controller can be found in the in Figure B.1.

The controller presented in [41] receives as inputs the error at a given discrete time t, $e(t)$, and the error derivation, defined as $\vec{\Delta e}(n) = \vec{x}_{\text{ref}}(t) - \vec{x}(t-1)$, where $\vec{x}$ denotes a given state. Hence, the controller behaves as a fuzzy logic PD controller. In total, there are three states, $\vec{x}$, tracking the desired reference values, $\vec{x}_{\text{ref}}$. The considered states are the drone's inertial coordinates relative to the drone's initial position, denoted as (x, y, z). Consequently, the controller receives as inputs three $(e, \Delta e)$ pairings which are then processed by the fuzzy logic units.

Regarding the fuzzy logic units, three are present (one per state). Each unit gets as inputs two values: e and Δe, with respect to the reference considered. On all fuzzy logic units, three linguistic terms (”Zero”, ”Positive, ”Negative”) are used to address the values of e and Δe. Each linguistic term is modeled using a Gaussian membership function, $G(c, \sigma)$ characterized by two parameters: the mean, denoted as c, and the standard deviation, σ. Furthermore, the antecedent terms all include the connective ”and” between the two inputs. Thus, a generic rule could be, for instance ”if e is Positive and Δe is Negative, then $y = f(e, \Delta e)$)”. As for the controller's consequent terms, since the controller is a first order TS, the output consists on a linear combination of the inputs, plus a bias. Consequently, in a given rule k, the consequent term takes the form of $y_k = g_{1k}e + g_{1k}\Delta e + b_k$, where g_{ik} represent the gains and b_k the biases.

In what concerns adaptability, both the antecedent and the consequent terms are subject to changes. Nonetheless, the number of rules in the fuzzy logic unit is fixed. Each rule contains a total of seven adjustable parameters. Concerning the antecedent term, the adjustable parameters are the standard deviations and expected values of the Gaussian membership functions for e and Δe. As for the consequent term of a given rule k the adjustable parameters consist in the gains g_{ik} and the biases b_k.

In E. Yazid's research, four evolutionary algorithms were used to tune the initial FLC (GA with two mutation rates, PSO and ABC), resulting in four distinct FLCs. These controllers were subsequently used to follow different kinds of reference functions, including multiple step functions and sinusoidal functions. Figure B.2 showcases the tracking of a sinusoidal reference on the x-axis whereas Figure B.3 shows the tracking of multiple steps. From the results presented, it can be seen that when tracking a sinusoidal reference all the tested algorithms yielded a similar and satisfactory performance. However, the tracking of a multiple step function shows that despite all final controllers being able to track the reference signal, the controller tuned using ABC exhibits a better performance when compared to the remaining approaches. This fact stresses that when testing evolutionary algorithms for TS controller tuning, different algorithms should be tested, as the difference in performance may be significant. Finally, each algorithm's final membership functions, together with a 3D-maping of the controller output computation are showcased in Appendix B (Figures B.6 and B.7).

Overall, the research in [41] suggests that adaptive FLCs, in particular, TS controllers, present a precise tracking of a variety of reference functions, including both step functions and sinusoidal ones. Furthermore, by analysing the changes in the FLC's membership functions, depicted in Figure B.6, it can be seen that in most cases, the membership functions move significantly from the initially defined positions, while often overlapping with each other. Lastly, there does not appear to be any pattern in the way the final membership functions are structured, such as a constant standard deviation or spacing between membership functions. In fact, these two traits vary significantly for both e and Δe, depending on the evolutionary algorithm used. This can be interpreted as one advantage of using this kind of optimization strategies, which enables finding non-intuitive (yet optimal) shapes for membership functions that could not be reached using a knowledge-based approach alone.

3.8.2. UAV RL-based Pursuit Evasion Game Controller

The second example discussed is presented by E. Camci et at. in 'Game of Drones: UAV Pursuit-Evasion Game With Type-2 Fuzzy Logic Controllers Tuned by Reinforcement Learning' [6]. As suggested by its name, the main goal of the simulations conducted consists on training a single pursuer

quadcopter (i.e. the controlled vehicle) to catch an evader quadcopter in the shortest possible time, while the evader strives to escape from the pursuer. The two quadcopters are different from each other in terms of their specifications such as mass, dimension or maximum speed. The main scenario of the game is that the pursuer starts from origin while the evader starts at a reasonable distance from it. Similarly to the first showcased example, all the relevant Figures are inserted in the Appendices, in particular in Appendix C

The article introduces a model-free RL based algorithm control strategy which is implemented in a simulated environment considering highly nonlinear quadcopter drones presenting coupled dynamics. It is additionally assumed that the modeled drones have to operate under noisy conditions. The control loop used can be found in Figure C.1. The controller chosen consists in a type-2 fuzzy logic TS controller, which works in a similar way to the type-1 TS controller presented in Section 3.2.2, with the main difference consisting in its membership values being defined as intervals. According to the authors, type-2 FLCs are considered better-suited to deal with high levels of uncertainty in the interpretation of input linguistic terms (e.g. when considering a significant amounts of noise) [6].

The controller's inputs consisted on the error, e and its time derivative $\dot{e}$. The error is defined as the distance between the pursuer's current position and the target's current position, expressed in (x, y, z) coordinates using an inertial reference frame. Regarding the fuzzy membership functions used in the antecedent term of the fuzzy logic unit, just as in the previously discussed article, these consisted in Gaussian membership functions, $G(c, \sigma)$. As for the consequent terms, the output function of every rule was defined as a constant, i.e. for a given rule n, $f_n = C$ with $C \in \mathbb{R}$. Furthermore, the connective "and" was present in all rules, regarding the two inputs $(x_1, x_2) = (e, \dot{e})$. Consequently, a generic rule n is given by:

$$\text{Rule}_n: \quad \text{IF} \;\; x_1 \text{ is } \tilde{A}_1^n \;\; \text{and } x_2 \text{ is } \;\; \tilde{A}_2^n \;\; \text{THEN} \;\; f_n = C_n$$

where $\tilde{A}_1^n$ and $\tilde{A}_1^n$ denote the type-2 fuzzy sets. Overall, three type-2 fuzzy set membership functions were implemented per input, which resulted in a total of 9 rules.

In what concerns the adaptive algorithm chosen to tune the controller, it follows a variant of the actor-critic approach presented in Section 3.7.1. The main difference from the previously presented approach consists in the estimator of the action-value function being an additional Fuzzy Inference System (FIS) instead of an ANN. Nonetheless, the FIS is tuned using a gradient descent approach, as described in Section 3.6.1, and exploration is considered by adding Gaussian noise to the default control action. This approach was chosen to avoid the discrete representation of such a large continuous space, which would be needed in case a classical RL approach, such as Q-learning, was used.

Furthermore, the reward function was defined as follows:

$$r_{t+1} = \frac{D_t - D_{t+1}}{(V_{\text{pursuer}} + V_{\text{evader}})T}$$

where D_t is the distance between the two drones at time instant t while T is the sampling time.

From the reward function definition it can be seen that positive rewards result from decreasing the distance between the pursuer and the evader (expressed in the numerator) in consecutive time steps and by decreasing the overall distance travelled by both drones in one sampling interval (expressed in the denominator).

In addition, it should be noted that only the consequent parts of the FIS and the type-2 TS-FLC are tuned, meaning that the membership functions maintain their respective initial parameters throughout the simulation process.

The drone controller was tested in several scenarios, with the two considered to be the most prominent showcased in Figures C.3 and C.4. In both scenarios the pursuer was modelled to be faster than the evader and started all the simulations at the origin. The values presented in the table referring to the first scenario (Figure C.3) show that for all the simulated initial conditions (i.e. for all the evader's initial positions), the pursuer managed to successfully catch the evader in a short amount of time. Additionally, it can be seen that the learning procedure significantly decreased the capture time for all the considered initial conditions. In the second considered scenario, depicted in Figure C.4, the antecedent membership functions were changed. Consequently, the second scenario aimed to test whether only tuning the consequent terms is sufficient to guarantee an increase in performance, even if the membership functions considered were not optimal. By looking at the results presented in the table referring to

the second situation, although the pursuer finds the evader in all the cases considered, it is evident that the tuning process did not significantly alter controller's performance. In fact, in one of the presented cases, the controller's performance is even significantly hindered by the tuning process, which takes 23.15 seconds to reach the target instead of the initially obtained 14.84 seconds.

From the previous points two main conclusions are inferred. Firstly, it can be stated that adaptive methods focusing solely on adjusting the consequent terms of FLC can produce significant changes in controller performance, as was shown in the first considered scenario. Nonetheless, controller performance also relies on the existing antecedent terms. In order to effectively boost performance by using the adaptive mechanism, the antecedents must be adequately chosen. This implies one of two scenarios: either there is enough knowledge available to accurately declare constant membership functions, or, alternatively, these functions need to be adapted as well, as was the case in the previously analysed paper. Secondly, the results presented show that even when having a small number of rules in a complex scenario, a satisfactory and comprehensive performance can be attained, with the pursuer always being able to catch the evader, before and after the tuning process. Additionally, it should be noted that positive results obtained in a scenario where a small number of rules is defined can be due to the dynamics defined in the simulation environment used to adapt the controller. In case the dynamics are not modelled as highly non-linear, then a controller with a small number of rules should be sufficient to handle any non-linearity involved in the process. Nonetheless, in this particular example, the good performances obtained conditions may also be due to the adaptiveness of the controller implemented. This implies that an adaptive approach for a TS controller holds great potential to deal with severe non-linearities and coupled dynamics and should be further considered.

3.8.3. Autonomous Vehicle Control Using Fuzzy Controllers Adapted using Reinforcement Learning

The third and final article discussed in the current Section is titled 'An Approach to Tune Fuzzy Controllers Based on Reinforcement Learning for Autonomous Vehicle Control", by X.Dai et al. [10]. The architecture of the controller proposed in this research is comprised of a Q Estimator Network (QEN) and a Takagi-Sugeno-type FIS. Consequently, as was the case in the previous described article, the proposed controller follows an actor-critic approach. Furthermore, the performance of the proposed design technique is tested by simulation studies of a vehicle longitudinal-control system. Just as in the previous examples, all the relevant results obtained in this research are included in the Appendices, in particular in Appendix C.

The controller proposed in [10] has two main tasks: estimate the optimal action-value function $Q^*(\vec{x}, a)$ (critic) and compute the control output based on the estimated action value function (actor) $\hat{Q}^*(\vec{x}, a)$. Additional information concerning the controller's block diagram can be found in the Appendices section, Figure D.1.

In order to achieve the first task, an ANN is used, whereas the second one relies on an FIS (TS controller). Concerning the QEN, a time-delay neural network (TDNN) structure is adopted. The TDNN is a backpropagation neural network that uses a time-delayed sequence as its input vector. This allows it to deal with input data that are presented over time. A schematic representation of the TDNN network format can be found in Figure D.2. In terms of the algorithm used to tune the TDNN, a gradient descent approach is implemented, explicitly expressed in (3.35).

Regarding the actor module, it consists on a TS controller where the consequent term is a linear combination of the inputs, plus a bias. Furthermore, all the membership functions considered in the antecedent term are Gaussian membership functions, $G(c, \sigma)$, and the number of rules is fixed (in the example provided, a total of nine were used). In the procedure proposed by X. Dai et al. [10], a gradient-descent method is used to tune the actor, aiming to maximize the action value function $Q(\vec{x}, u)$, with respect to the control input u for the current state $\vec{x}$. The parameters tuned by the gradient descent approach were both the antecedent's membership functions (c and σ) and the consequent's coefficients. In addition, it should be noted that the initial parameter values of the membership function values were not set randomly: the user implements initial knowledge into the control system rule-base, which is subsequently tuned by the RL approach. Exploration is taken into account by adding a stochastic action modifier (i.e. adding Gaussian noise) immediately before the control input is sent to the system. Moreover, the noise's standard deviation, σ_n, converges to zero gradually, meaning the policy becomes greedy with time.

The proposed controller was applied for a vehicle longitudinal-control problem. This problem addresses the control of a vehicle to maintain a safe distance between the controlled car and the preceding cars. In the simulation presented, there are four input-state variables available in this system: the position of the preceding car x_l, the velocity of the preceding car v_l, the position of the following car x_f, and the velocity of the following car v_f. The proposed controller adopts reinforcement learning to tune the fuzzy controller, so it is a model-free paradigm that is capable of learning the optimal strategy through trial-and-error interactions with a dynamic environment. The input variables to the TS controller are the spacing deviation $\Delta x = x_l - x_f$ and the relative speed $v_r = v_l - v_f$. The reward function was defined as follows:

$$r(t) = \begin{cases} -\frac{1}{6}|\Delta x|, & |\Delta x| \leq 6 \\ -1, & |\Delta x| > 6, \text{end of episode.} \end{cases}$$

As for the QEN, its inputs consisted of the three most recent values of Δx, the three latest values of v_r and, lastly, the control input, u. The obtained results are illustrated in Figure D.3 and D.4. Figure D.3 shows three velocity profiles: the ideal velocity profile (equal to the leading car's velocity profile) and the velocity profiles of the controlled car before and after learning takes place. Figure D.4 showcases the evolution of the spacing deviation Δx in time.

From the simulation results shown in Figure D.3, it can be seen that the controlled car is able to nearly match the ideal velocity profile, after the learning process takes place. The two velocity profiles are not identical due to two reasons: firstly due to the sudden velocity changes in the preceding car (represented by the ideal velocity line) and, secondly, because of the small number of fuzzy rules used in the simulation. Additionally, concerning the spacing deviation graph, the values of Δx can be seen to decease gradually. During the learning procedure, the maximum spacing deviation may be increased, this is due to the implemented exploration strategy. Nonetheless, it can be seen that, as a whole, the maximum spacing is decreased. Lastly, the final spacing deviation is not zero due to the small number of fuzzy rules implemented.

Overall, the approach taken was considered to yield satisfactory results, as the RL tuning process enabled the controller's performance to increase significantly. Furthermore, it is considered that the performance could be increased even further by increasing the number of rules present in the actor's fuzzy rule-base. Nevertheless, it should be taken into account that, according to the authors in [10], it has been demonstrated that simply replacing the discrete lookup tables with function approximators may be not robust and cause learning to fail, even in benign cases. Consequently, this method may not be as reliable in more complex scenarios as it is in simpler problems. Lastly, the fact that all the articles presented so far rely on using Gaussian membership functions for the antecedent terms' membership functions (mostly due to these only exhibiting two adjustable parameters per membership function) can be seen as an indicator that adopting a similar strategy when developing an FLAC might lead to a better performance after controller tuning and thus should be further considered.

4

Controller Design for a SAD

In the previous Chapter 2, the concept of SAD was introduced, together with the traits considered to be fundamental for a successful therapeutic implementation. Subsequently, in Chapter 3, several state of the art control methods were mentioned, along with experimental results associated with the proposed methods. The current section will take into account the information provided in the previous two and apply it to the scope of SADs. In other words, it will regard controller development for SADs.

4.1. Drone Traits Influenced by the Controller

Chapter 2 listed the five traits that hold the most importance when considering a SAR. These were: appearance, simplicity, adaptability, responsiveness and autonomy. Nonetheless, not all of these are influenced by the controller. For instance, appearance is completely dissociated from it. Regarding responsiveness and simplicity, these traits are mainly influenced by the interface presented to the therapist and not as much by the controller itself. In other words, as was mentioned in Chapter 3, the best approach to create a responsive and simple SAD is to focus on developing an intuitive and comprehensive GUI. This would enable the therapist to implement changes in the SAD's behavioural patterns in a fast and simple manner, thus ensuring an appropriate level of responsiveness and simplicity.

However, the remaining two traits, adaptability and autonomy fully rely on the controller developed. Unlike responsiveness and simplicity, the previous two traits are not directly related to the information and interface provided to the therapist. Instead, these are solely linked to the SAD's behaviour. Concerning autonomy, it influences the SAR's behaviour by ensuring that the control system performs accurately without external intervention and, regarding adaptability, it is responsible for continually adapting the performance in pursuit of a personalized approach for each patient.

Consequently, the following analysis will take into consideration which type of control strategy is the most adequate to provide the required levels of adaptability and autonomy in the context of SADs.

4.2. Achieving Autonomy: Fuzzy Logic Controllers for SADs

When considering the therapeutic applications of an SAD, particularly the fact that it will be participating in therapy sessions with children with ASD, several performance requirements for the implemented controller come to mind. Firstly, since a DMT approach is easily gamified (that is, presented in the format of an interactive game), an approach enabling the implementation of a priori knowledge into the system (e.g. game rules) would be preferred. Additionally, due to the fact that DMT approaches are based on human-interaction, coding the controller in such a way that the drone presents a humanlike behaviour would also be beneficial. In this case, humanlike behaviour refers to the drone's ability to react in a similar way to a human, based on intuitive rules instead of mathematical models. Furthermore, in order to ensure safety for all participants involved in a therapy session, high performance is needed. This would ensure that the controller, and hence the SAD, would perform effectively in the widest range of possible scenarios.

In light of the listed features, and taking into account what was mentioned in Section 3.2.3, it is considered that an FLC is the indicate choice for the task. When compared to the traditional approach, the fact that FLCs do not require any mathematical model for the external process stands out as a key asset of this type of controller. Due to the constant interactions with humans, whose behaviour can hardly be modeled or predicted, setting a simple yet comprehensive rule base to determine the controller's output seems like the most adequate choice to provide the desired levels of autonomy.

4.3. Achieving Adaptability: Adaptive Control for SADs

In Chapter 2, the main benefits of presenting a high degree of adaptability in a SAR were discussed. These included increasing the interest displayed by patients when interacting with SARs and the number of therapy sessions in which these could be useful for the patient. In practice, a high level of adaptability is achieved through a personalized approach. In other words, it is achieved by making sure that the SAR behaves differently with each patient and presents the ability to continually adapt its behaviour to cover each patient's individual needs. It is considered that the same principles hold when concerning an SAD.

In terms of controller development, and taking into account what was mentioned in Section 3.3, the conclusion is evident: an adaptive approach must be taken. From all the alternatives discussed in Chapter 3, I will focus primarily on those referred to in Section 3.8 due to two factors. Firstly, all the examples described in Section 3.8 address the issue of tuning an FLC, which, given all that was stated up to this point, has already been established as the most convincing approach to take. Secondly, these approaches all rely on state of the art control methods, such as ANN and RL. Consequently, it is considered that developing a controller based on these methods should lead to better final results in terms of controller performance. In addition, using state of the art methods is also considered to increase the inherent value of this research project in terms of contributions to the current body of knowledge.

Still concerning the three research papers presented in Section 3.8, two different approaches for tuning FLCs were presented. In the first article an evolutionary approach was taken, whereas in the following two the tuning process consisted in using an RL-based approach. It is considered that the latter approach is better suited for the specific case of tuning a controller for an SAD. This statement is based on a significant drawback associated with genetic algorithms: the need for developing a cost function responsible for quantifying how well a controller with a given set of adjustable parameters performs. In the case-study presented by E.Yazid et al. [41], controller performance was measured based on how well the controller could track a set of references including steps and sinusoidal functions. Nonetheless, when considering an SAD, controller performance is not as easy to quantify since the main goal of the SAD is not to accurately follow references, but instead to interact as much as possible with a given patient. Finding a function or an alternative method which would be able to quickly quantify the controller's performance based on a current set of adjustable parameters is not intuitive. Consequently, using an evolutionary approach, which continually relies on the existence of such a function or method to compute the optimal set of controller adjustable parameters, does not stand as a feasible answer to the problem.

On the other hand, the RL-based approaches detailed in Sections 3.8.2 and 3.8.3 seem significantly more intuitive. By means of defining an appropriate reward function and by being able to develop a simulation environment, it should be possible to tune the FLC in such a way that it is able to perform DMT while, for instance, always keeping a safe distance from the patient (e.g. attributing positive rewards when the SAD's distance to the human user is adequate). In this scenario, the main challenge will be choosing the adequate reward function.

Lastly, one additional detail is considered. Between the algorithms presented in Sections 3.8.2 and Section 3.8.3, there are two main differences. Firstly, in the research conducted by E. Camci et at.in [6], only the consequent term of the TS-controller was adaptively tuned, whereas in the procedure followed by the X. Dai et al.[10], both the consequent coefficients and the antecedent membership functions were adaptively tuned. Between these two approaches, the second one seems the best fit for an SAD, as it is more comprehensive and, consequently, should be able to adapt the controller further, achieving a higher degree of personalization. Additionally, by being able to change the membership functions defined a priori by the programmer, the second approach also reduces the subjectivity associated with each rule (and, in particular, with each linguistic term used) to some extent. The second main

difference between the two approaches is the fact that the first one uses a type-2 fuzzy logic controller. This approach is followed in order to better deal with high levels of noise in the input. In case of the SAD, the input is not expected to be noisy enough to justify the increase of complexity associated with type-2 fuzzy logic systems. Therefore, a type-1 FLC will be used.

With the purpose justifying the previous assumption that the input will not include significant noise, the inputs and outputs of the controller will now be described.

4.4. Controller Inputs and Outputs: Using ANN for Image Processing

In order to perform DMT, the SAD needs to regularly check the human user's position, emotional status, and status of interaction with the SAD. More specifically, the SAD needs to analyse how well the patients are mimicking the movements suggested to them. In addition, it needs to figure out which new movements to propose to the human user in case of active interaction or, alternatively, which of the human user's moves to mimic in case of passive interaction. This implies that the SAD must integrate an IPU. The development and implementation of the IPU is considered to be out of the scope of the present thesis project. Therefore, for future reference, it is assumed that the controller inputs are generated by the IPU, i.e. that information such as proximity to the patient or the position of its limbs is available and accessible to the controller at any time, serving as its inputs. Due to the camera quality of the Parrot bebop drone and the efficiency of state of the art image processing algorithms, the inputs are not expected to be noisy, which further grounds the choice of a type-1 FLC.

Regarding its outputs, the controller for the SAD will control the drone by directly setting reference pitch and roll angles as well as the yaw rate. Further details regarding the controller outputs will be presented in the following Chapter 5.

4.5. Conclusions

In conclusion, considering all the previous discussions, the ideal controller for an SAD meant for usage in autism therapy sessions is thought to be a type-1 FLC tuned by DRL algorithms. This controller architecture is chosen due to it being the architecture which guarantees the highest levels of personalization to each patient's individual needs (high adaptability), while simultaneously providing a comprehensive (and thus autonomous) way to conduct DMT. In order to achieve this, in an initial phase, a simple FLC will be tested relying solely on the face detection feature provided by the IPU. During this first phase, simple interaction patterns will be tested. In a secondary phase, new inputs such as the position of the human's limbs will be additionally considered in order to implement new, more complex, interaction patterns. Lastly, an adaptive module will be implemented and the final controller's performance will be tested and compared to that of the initial stages. The following Chapter 5 thoroughly describes the implementation of the first phase, along with some of the limitations and problems faced in the early stages of controller development.

5

Preliminary Results

The present Chapter describes the procedure conducted so far regarding the design of the controller and implementation of the developed controller for an SAD.

5.1. Experimental Setup & Software

During the first stage of the experimental procedure, a simple non-adaptive FLC was implemented on a Parrot Bebop 2 drone. This controller was developed in Python 3.6 and sent instructions to the drone via wi-fi by making use of the Pyparrot library (based on Parrot's official software, Olympe) [23]. The FIS was implemented using the PyFuzzyLite library [29]. All tests were conducted at the Cyberzoo drone testing facilities at TU Delft's faculty of aerospace engineering. Furthermore, all human-drone interactions presented assume that only one person (the author) is interacting with the drone. Cases involving multiple human users are disregarded. In addition, drone motions were restricted to the (x, y) plane and a small control sampling time of 0.05 seconds was used between introducing different control inputs to the SAD.

5.2. Controller Inputs and Outputs

The controller receives as inputs the output of a pre-implemented IPU. This IPU consists in an ANN responsible for identifying and locating human faces in real time video footage, with its output being the coordinates (in pixels) of rectangles containing the faces of any human appearing on the image. The IPU's output will henceforth be referred to as the facebox value or controller input. As for the controller's output, it consists in the desired pitch and roll angles and the desired yaw rate of the drone. Due to the fact that the actuators are not directly controlled, the controller developed is an outer-loop controller.

5.3. SAD Game Modes

In order to test the drone's performance in multiple types of scenarios involving human interaction, three Mamdani controllers were developed. Furthermore, to ensure a smooth transition between the controllers, a navigational algorithm was created. Each controller was associated with a specific task, referred to as game mode. Subsequently, three game modes were tested: Stand by, Pursuit and Mimic. Figure 5.1 summarizes the interactions between the three game modes.

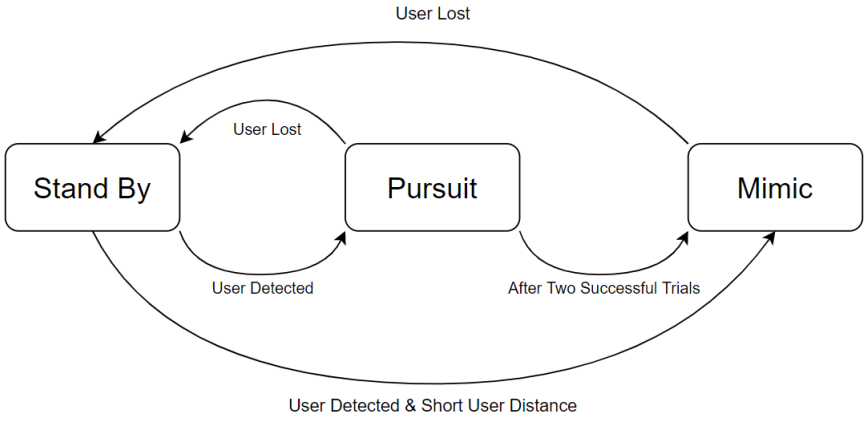


Figure 5.1: Control Algorithm Scheme

5.3.1. Stand By Game Mode

The so-called Stand By mode has as its main function to find the human user. In order to achieve this, the drone only changes its yaw angle, ideally remaining in the same position while rotating. User detection is conducted by continually checking the facebox input value. Once a human user is detected, the drone ensures that the user's facebox is centered within the captured images of the drone by continually checking the location of the facebox's centroid (tolerance was set to 20% of the image length, in pixels). Once the human user (or, equivalently, the user's facebox) is centered on the image captured by the drone, the control algorithm updates the current game mode from Stand By to either Pursuit or Mimic. Figures 5.2 and 5.3 showcase the Stand By game mode before and after centering takes place.

Figure 5.2: Stand By game mode before centering: the drone is trying to find the human user's facebox by rotating around a fixed axis

Figure 5.3: Stand By game mode after centering: the drone is directly facing the human user and keeps its position and heading until the user (or equivalently the user's facebox) starts to move

5.3.2. Pursuit Game Mode

The second game mode was designed to simulate a game of catch between the drone and the human user. In other words, it attempts to significantly reduce the distance between the two, while maintaining a safe distance from the human user, as well as maintaining the user's face centered in the image. In order to do so, pitch and roll are controlled by the FLC. In this second mode, the yaw rate is no longer controlled as whenever this mode is active, the user has already been identified within the image captured by the drone and consequently, controlling pitch and roll is enough to center the user's face in the image. The distance between the drone and target (i.e. the human) is estimated from

the area of the facebox: the bigger the area, the closer the human user. The controller thus tries to achieve a facebox ratio (defined as the facebox area divided by the total amount of pixels present in an image) that ensures that the human user is close enough to the drone and yet not too close to raise any safety issues. Furthermore, again to ensure safety, the maximum pitch and roll angles were significantly limited. By restricting the maximum pitch and roll angles, the drone's maximum velocity (i.e. both its longitudinal and lateral components) is reduced, guaranteeing that the drone never approaches the human with excessive velocity. After achieving a satisfactory facebox ratio (less than or equal to 4%), which was heuristically tuned, the first stage of the Pursuit game mode is considered finished. Subsequently, the drone waits a certain amount of time and a second (and final) trial identical to the first one begins. If at any point in time the human user is no longer detected by the IPU, the drone reverts to Stand By mode and attempts to once again locate the human user.

Figures 5.4 and 5.5 present a scenario before pursuit starts and once it is completed, with the human user being allowed to move throughout the process.

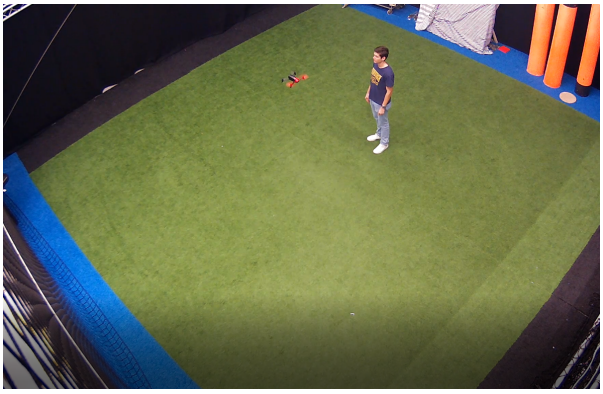


Figure 5.4: Pursuit game mode initial positions of the drone and the human user

Figure 5.5: Pursuit game mode when the moving human user is reached by the drone

5.3.3. Mimic Game Mode

The third and last game mode developed is aimed at motivating the human user to move according to the drone's movements. The movements consist on slight motions either to the left or to the right. After the drone finishes a motion, it remains in the final position for a certain amount of time (i.e. 5 seconds in the performed experiments), waiting for the human user to mimic its movement. After this time has passed, the performance of the human user is assessed and the result is stored. User performance is assessed by the drone using a performance metric which can take three values: 0 in case the user does not follow the drone at all (and consequently no facebox is detected); 0.5 in case the user completes the motion, but is not aligned with the drone (i.e. the facebox's centroid is not centered in the image captured by the drone given a tolerance threshold which, in case of the performed experiments, is 15%); and, lastly, 1 in case the user accurately moves along with the drone (i.e. the facebox is considered centered at the end of the time interval). After performance assessment, the drone repeats the procedure two additional times and lands at the end. The FLC's rule base is implemented in such a way that the amplitude of the movements performed by the drone increases with the human user's precision in mimicking the movements, in other words, the more centered the facebox centroid after each motion, the bigger the amplitude of the following motion.

Figures 5.6 and 5.7 showcase the drone's position before and after a standard move to the sides takes place.

5.4. Limitations

From the experimental procedure conducted, one main limitation was identified: the input considered was only accurate when the human user was within a two meter radius from the drone. In case the user was located beyond this threshold, the drone proved to be unable to recognize the user's facebox

Figure 5.6: Mimic mode initial positions

Figure 5.7: Mimic mode after drone movement

and, thus, it could not interact with the user. Additionally, the current controller inputs do not enable identifying any physical barrier which might obstruct the drone's movements. Once again, this fact raises safety issues regarding usage in indoor locations.

5.5. Future work

The next steps to be implemented will focus, on the one hand, in addressing some of the limitations faced, and, on the other, in implementing additional functionalities aiming to improve the drone-user interaction. In particular, the ANN used for user recognition will be enhanced in such a way that will enable recognition of some of the user's joints. Consequently, the joint coordinates within the image, in pixels, will be provided to the controller as an additional input. This will allow to overcome the 2 meter radius liability explained before.

Furthermore, so far, within the Mimic Game Mode, the drone must always start the interaction in hope that the human user mimics it, leading him/her to always play a passive role. In the next stage, a new Game Mode will be devised, where the drone will be the one taking the passive role, waiting for the human user to move and responding by mimicking his/her movements with a similar speed and amplitude.

Lastly, an adaptive module will be implemented into the existing FLCs. This module is expected to yield an increase in personalization. To complement its implementation, a new TS controller architecture will be adopted instead of the current Mamdani architecture. The adaptive module will take into account parameters such as the user's performance in order to adapt the FIS and, consequently, the drone's movements.

6

Closing Remarks and Future Work

The research conducted up to this point has concerned a variety of topics. Firstly, the condition of autism spectrum disorder or ASD was introduced in a comprehensive way: from its idiosyncrasies to its socioeconomic impact. This provided context, purpose and motivation for the problem being addressed by this master thesis project: developing a controller fit for a novel, SAR-based therapeutic approach meant to hinder the impacts of ASD in those affected by the condition.

Subsequently, further research revealed the benefits of using dance movement therapy or DMT as an answer to the problem. The idea of associating DMT to the concept of SAR was considered as one of the main contributions of this project and an in-depth analysis of several SAR models reaffirmed the potential inherent to the DMT-SAR approach. The scenario of using drones as SARs, i.e. SADs, was then examined. Although its usage is linked to some drawbacks, these are considered to be outweighed by the potential benefits, such as their increased mobility and captivating appearance. Nonetheless, in part due to this increased mobility, implementing an adequate control system which guarantees key features such as autonomy, adaptability or responsiveness is crucial. As such, several control algorithms were thoroughly described, with an analysis being conducted on the ones which are considered the best fit for implementation in this project.

In light of this, two main points stand. First and foremost, it is believed that the problem addressed by this research is now known in a comprehensive and detailed way. As a result, the key features which must be incorporated in the developed controller have been clearly identified. Secondly, taking into account all the prototypes and technical analyses considered in Chapter 2 and Chapter 3, it is considered that the selected approaches present high chances of being successfully integrated into therapeutic sessions.

Regarding future work, our main focus will be on developing an adequate controller for an SAD, which is able to promote and sustain interactions with humans based on structured DMT plans. As was addressed in Chapter 5, in the first stages of development, standard FLCs were implemented to a Parrot Bebop 2 drone. This allowed identifying a few limitations which will be addressed in a later phase of this MSc thesis project, including the short range needed for user detection or the lack of a personalized behaviour. With the aim of addressing the latter, in future project stages, an RL-based adaptive module will be developed and implemented into the existing version of the controller. Additionally, a new set of controller inputs including human joint positions will be implemented. This will enable to increase the range of detection and, simultaneously, to develop new interaction schemes, where the user will take an active role. In the final research phase, the adaptive version of the controller will be implemented and tested on real hardware.

Appendices

Appendix A: Yearly Drone Market Investments

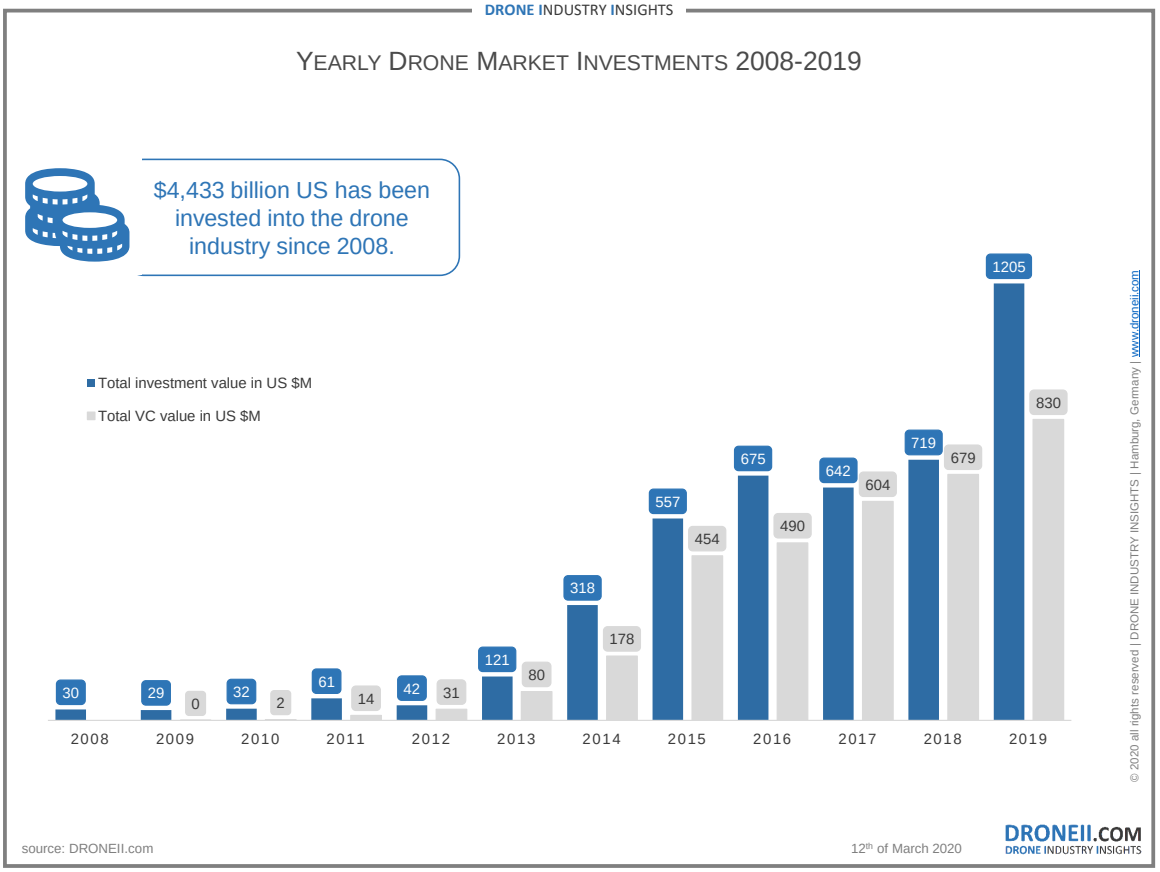


Figure A.1: Yearly drone marked investment between 2008 and 2019. Source: Droneii.com

B

Appendix B: Experimental Results 1

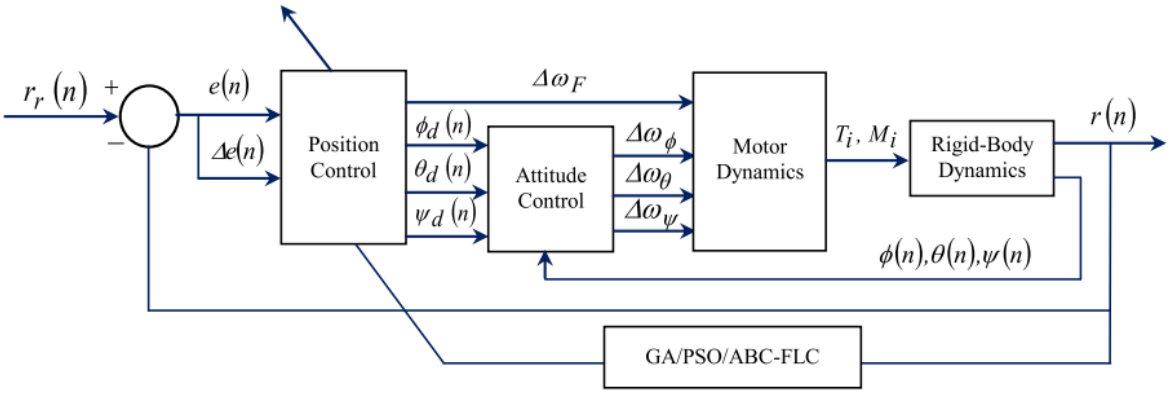


Figure B.1: Block Diagram Representing Control Scheme used by E.Yazid in [41]

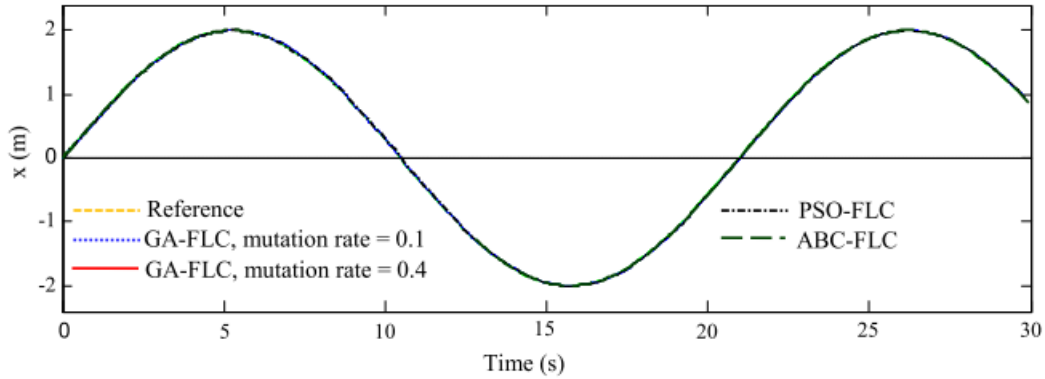


Figure B.2: Results obtained in [41] when tracing a sinusoidal reference on the x-axis

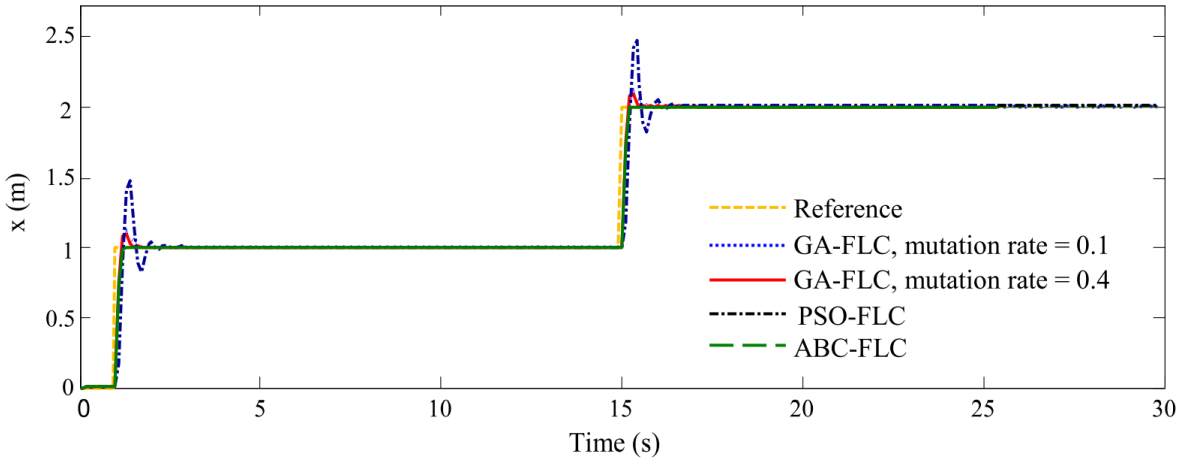


Figure B.3: Drone response for varying step function in x-direction in time window 0-30 s [41]

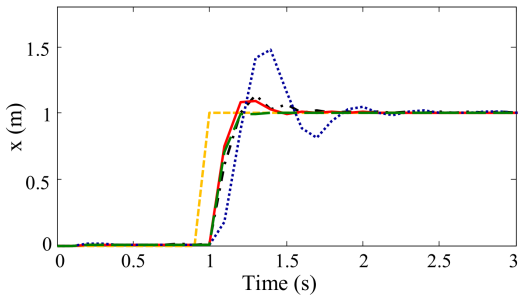


Figure B.4: Drone response for varying step function in x-direction in time window 0-3 s [41]

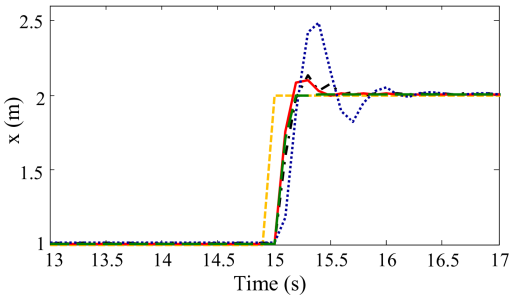


Figure B.5: Drone response for varying step function in x-direction in time window 13-17 s [41]

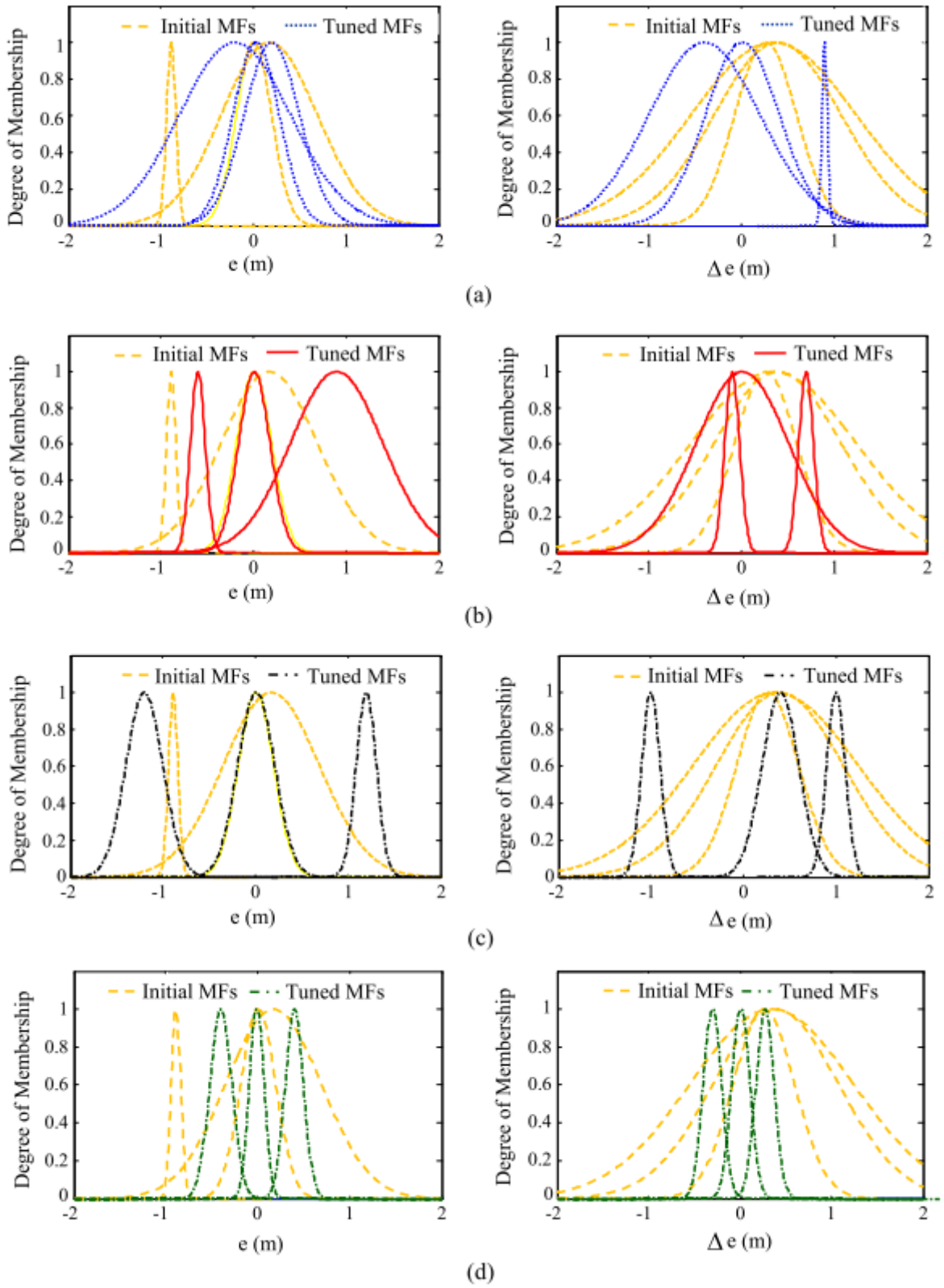


Figure B.6: Initial and final fuzzy sets of e and Δe for sine function in x-axis (a) GA-FLC with mutation rate = 0.1 (b) GA-FLC with mutation rate = 0.4 (c) PSO-FLC (d) ABC-FLC [41].

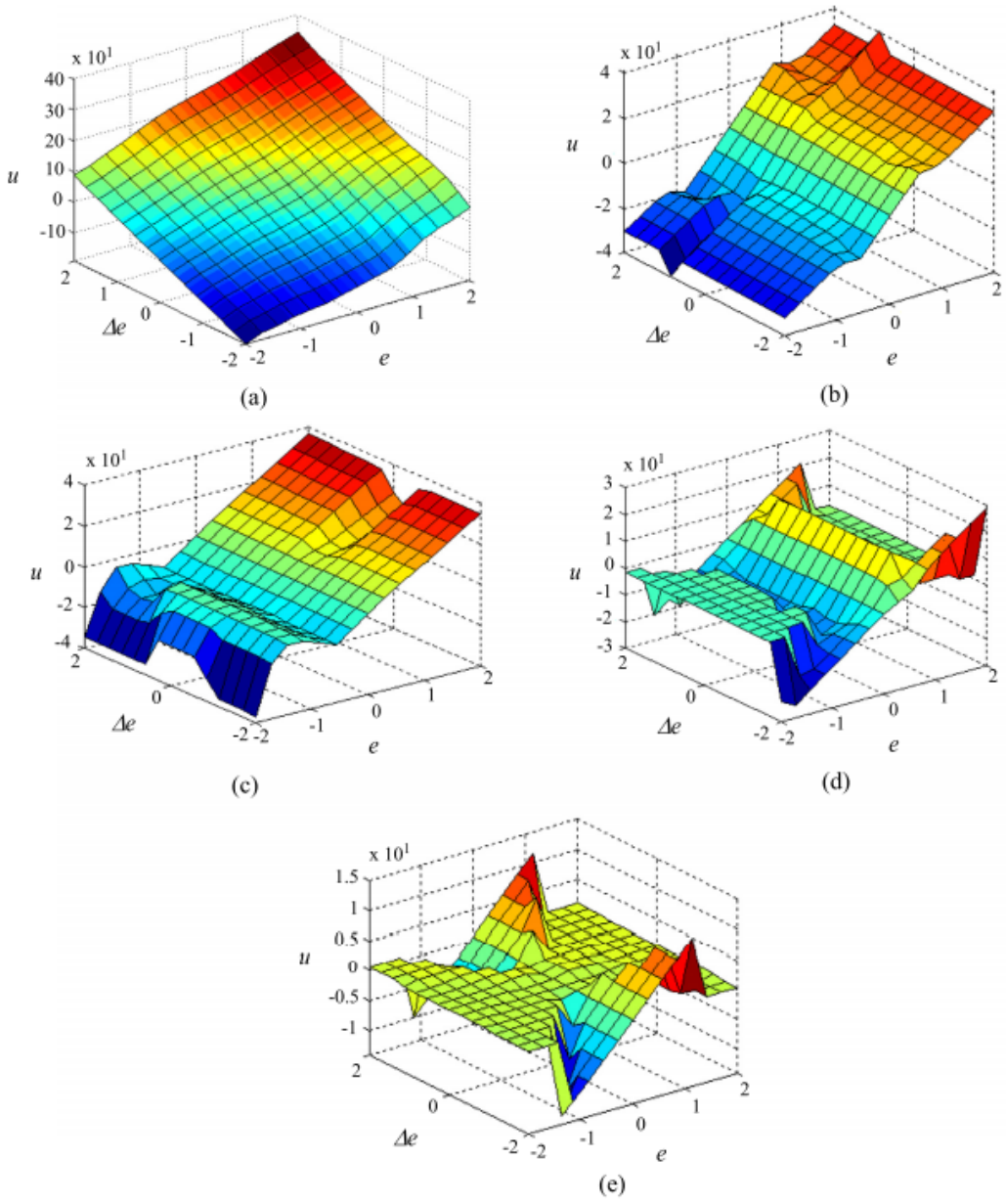


Figure B.7: Three-dimensional control surface of FLC for sine function in x-direction (a) initial (b) GA-FLC with mutation rate = 0.1 (c) GA-FLC with mutation rate = 0.4 (d) PSO-FLC (e) ABC-FLC [41].

C

Appendix C: Experimental Results 2

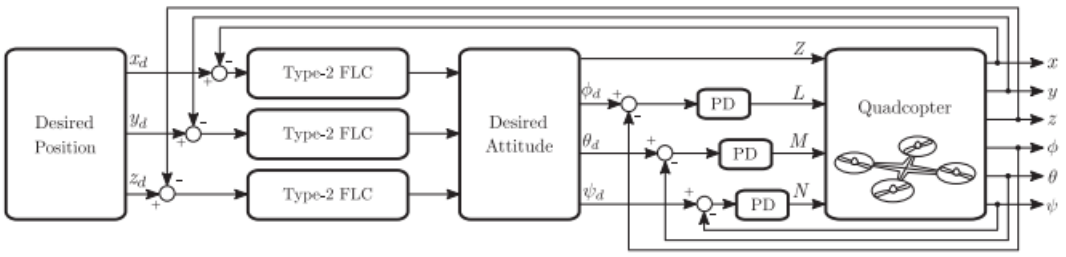


Figure C.1: Control architecture of the pursuer quadcopter in [6]

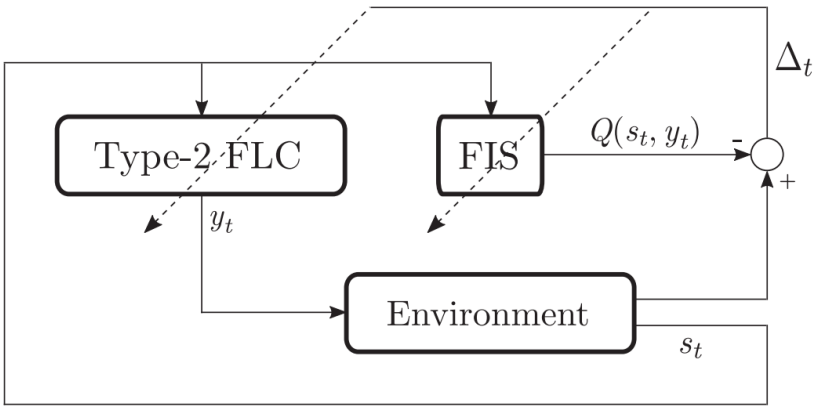


Figure C.2: Architecture of Learining used in [6]

Initial Position *(x, y, z)*	Capture Time (s)	
	Before learning	After Learning
$(14, 17, 15)$	40.31	30.34
$(11, 19, 13)$	13.49	12.83
$(19, 12, 16)$	32.25	20.81

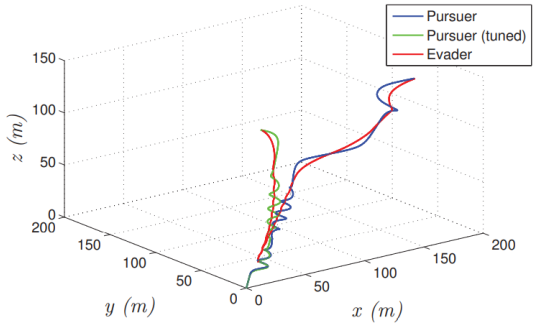


Figure C.3: Simulation results relative to the first scenario presented in [6]

Initial Position *(x, y, z)*	Capture Time (s)	
	Before learning	After Learning
$(20, 14, 12)$	14.84	23.15
$(19, 17, 18)$	28.83	27.02
$(15, 18, 17)$	30.02	28.08

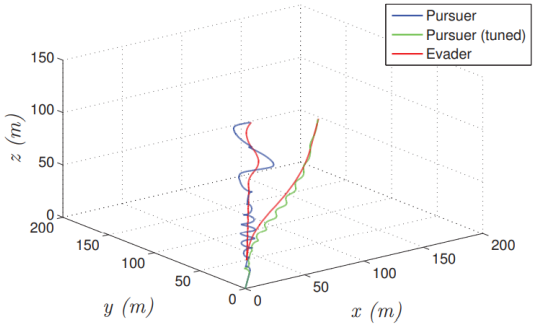


Figure C.4: Simulation results relative to the second scenario presented in [6]

Appendix D: Experimental Results 3

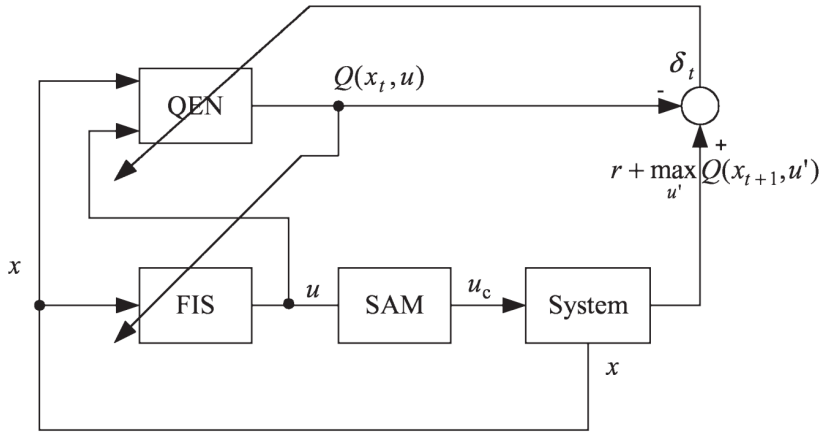


Figure D.1: Architecture of the controller proposed by X.Dai et al. in [10]

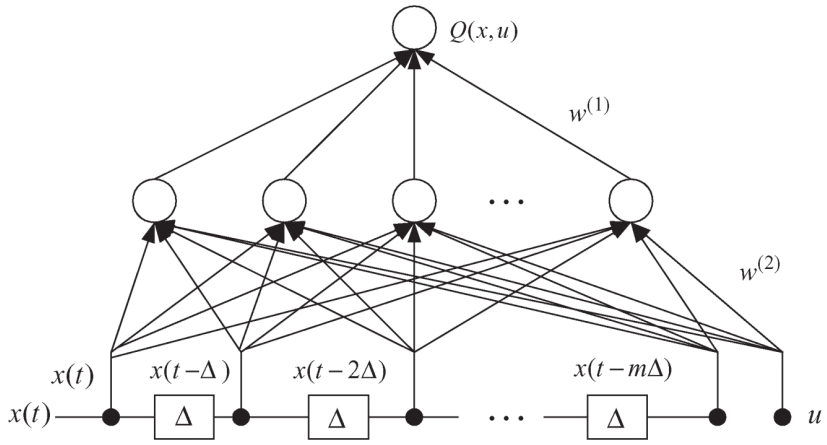


Figure D.2: Architecture of the QEN used by X.Dai et al. in [10]

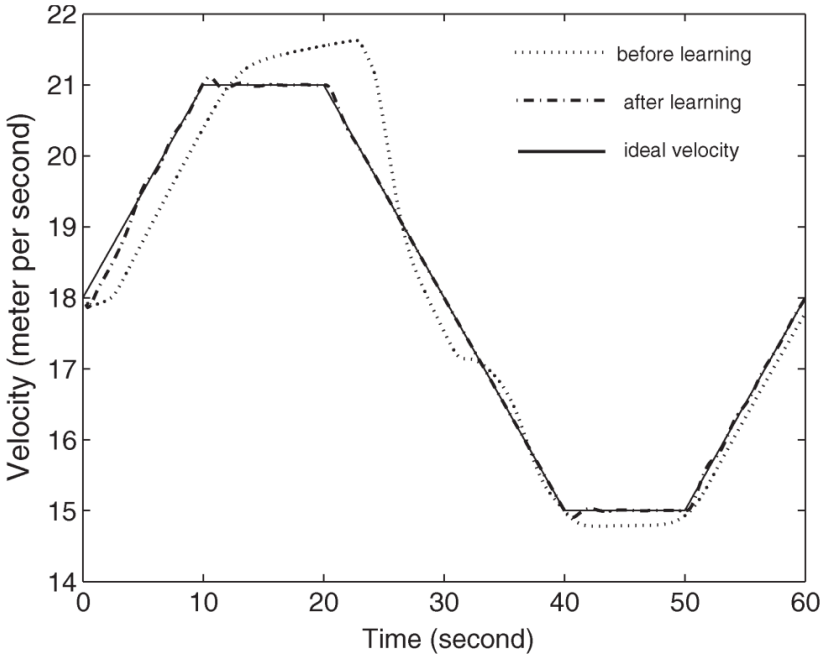


Figure D.3: Simulation results of the proposed controller: velocity response of the controller before and after learning takes place [10]

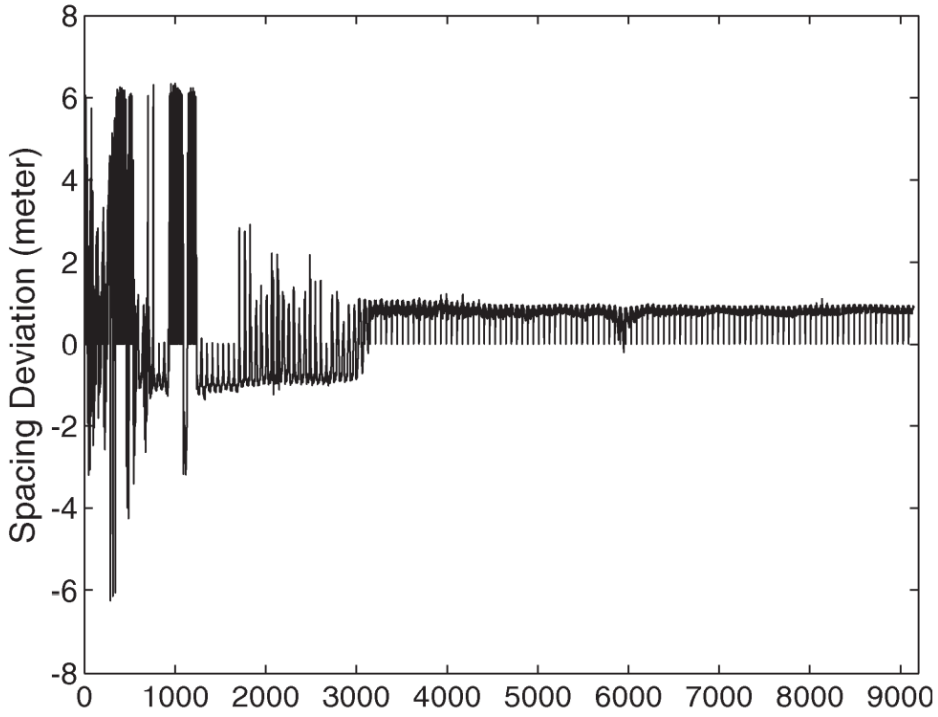


Figure D.4: Simulation results of the proposed controller: evolution of spacing deviation [10]

Bibliography

[1] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[2] Karl Johan Åström. Theory and applications of adaptive control: a survey. *Automatica*, 19(5): 471–486, 1983.

[3] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on Systems, Man, and Cybernetics*, (5):834–846, 1983.

[4] Yoshua Bengio. *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[5] Laura Boccanfuso and Jason M O'Kane. Charlie: An adaptive robot design with hand and face tracking for use in autism therapy. *International journal of social robotics*, 3(4):337–347, 2011.

[6] Efe Camci and Erdal Kayacan. Game of drones: UAV pursuit-evasion game with type-2 fuzzy logic controllers tuned by reinforcement learning. In *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 618–625, Vancouver, Canada, 2016.

[7] Zuleyha Cidav, Steven C Marcus, and David S Mandell. Implications of childhood autism for parental employment and earnings. *Pediatrics*, 129(4):617–623, 2012.

[8] Bart Custers. Drones here, there and everywhere: introduction and overview. In *The Future of Drone Use*, pages 3–20. Springer, 2016.

[9] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[10] Xiaohui Dai, Chi-Kwong Li, and Ahmad B Rad. An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems*, 6(3):285–293, 2005.

[11] Didier J Dubois, Henri Prade, and Ronald R Yager. *Readings in fuzzy sets for intelligent systems*. Morgan Kaufmann, 2014.

[12] Mayada Elsabbagh, Gauri Divan, Yun-Joo Koh, Young Shin Kim, Shuaib Kauchali, Carlos Marcín, Cecilia Montiel-Nava, Vikram Patel, Cristiane S Paula, Chongying Wang, et al. Global prevalence of autism and other pervasive developmental disorders. *Autism research*, 5(3):160–179, 2012.

[13] David Feil-Seifer and Maja J Mataric. Defining socially assistive robotics. In *9th International Conference on Rehabilitation Robotics. ICORR 2005.*, pages 465–468, Chicago, IL, US, 2005.

[14] Terrence Fong, Illah Nourbakhsh, and Kerstin Dautenhahn. A survey of socially interactive robots. *Robotics and autonomous systems*, 42(3-4):143–166, 2003.

[15] Nicole Giullian, Daniel Ricks, Alan Atherton, Mark Colton, Michael Goodrich, and Bonnie Brinton. Detailed requirements for robots in autism therapy. In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 2595–2602, Istambul, Turkey, 2010.

[16] Marcel Heerink, Ben Kröse, Vanessa Evers, and Bob Wielinga. Assessing acceptance of assistive social agent technology by older adults: the Almere model. *International journal of social robotics*, 2(4):361–375, 2010.

[17] R. Jager. *Fuzzy Logic in Control*. PhD dissertation, Delft University of Technology, 1995. Delft, The Netherlands.

[18] Martin Knapp, Renée Romeo, and Jennifer Beecham. Economic cost of autism in the UK. *Autism*, 13(3):317–336, 2009.

[19] Matthew J Maenner, Kelly A Shaw, Jon Baio, et al. Prevalence of autism spectrum disorder among children aged 8 years - autism and developmental disabilities monitoring network, 11 sites, United States, 2016. *MMWR Surveillance Summaries*, 69(4):1, 2020. Report published by Centers for Disease Control (CDC). DOI: http://dx.doi.org/10.15585/mmwr.ss6904a1.

[20] Eleni Mangina, Evan O'Keeffe, Joe Eyerman, and Lizbeth Goodman. Drones for live streaming of visuals for people with limited mobility. In *2016 22nd International Conference on Virtual System & Multimedia (VSMM)*, pages 1–6, Kuala Lumpur, Malaysia, 2016.

[21] Patrizia Marti, Margherita Bacigalupo, Leonardo Giusti, Claudio Mennecozzi, and Takanori Shibata. Socially assistive robotics in the treatment of behavioural and psychological symptoms of dementia. In *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006. BioRob 2006.*, pages 483–488, Pisa, Italy, 2006.

[22] Mary Martin. Moving on the spectrum: Dance/movement therapy as a potential early intervention tool for children with autism spectrum disorders. *The Arts in Psychotherapy*, 41(5):545–553, 2014.

[23] Amy McGovern. Pyparrot library, 2017. URL `https://github.com/amymcgovern/pyparrot`. Last Time Accessed: 27-02-2021.

[24] Raman Mehra. On the identification of variances and adaptive kalman filtering. *IEEE Transactions on Automatic Control*, 15(2):175–184, 1970.

[25] Christina Moro, Goldie Nejat, and Alex Mihailidis. Learning and personalizing socially assistive robot behaviors to aid with activities of daily living. *ACM Transactions on Human-Robot Interaction (THRI)*, 7(2):1–25, 2018.

[26] Katsuhiko Ogata. *Modern control engineering*. Pearson, 5th edition, 2010.

[27] World Health Organization. World health organization fact sheets. `https://www.who.int/news-room/fact-sheets/detail/autism-spectrum-disorders`, 2020. Last Time Accessed: 27-02-2021.

[28] J. Kober R. Babuška. Knowledge-based control systems, 2019. Delft University of Technology.

[29] Juan Rada-Vilela. The fuzzylite libraries for fuzzy logic control, 2018. URL `https://fuzzylite.com/`. Last Time Accessed: 27-02-2021.

[30] Daniel J Ricks and Mark B Colton. Trends and considerations in robot-assisted autism therapy. In *2010 IEEE international conference on robotics and automation*, pages 4354–4359, 2010.

[31] Ben Robins, Kerstin Dautenhahn, R. Te Boerkhorst, and Aude Billard. Robots as assistive technology-does appearance matter? In *RO-MAN 2004 13th IEEE International Workshop on Robot and Human Interactive Communication (IEEE Catalog No. 04TH8759)*, pages 277–282, 2004.

[32] Ben Robins, Kerstin Dautenhahn, and Janek Dubowski. Does appearance matter in the interaction of children with autism with a humanoid robot? *Interaction studies*, 7(3):479–512, 2006.

[33] Aine Roddy and Ciaran O'Neill. The economic costs and its predictors for childhood autism spectrum disorders in Ireland: How is the burden distributed? *Autism*, 23(5):1106–1118, 2019.

[34] Syamimi Shamsuddin, Hanafiah Yussof, Luthffi Ismail, Fazah Akhtar Hanapiah, Salina Mohamed, Hanizah Ali Piah, and Nur Ismarrubie Zahari. Initial response of autistic children in human-robot interaction therapy with humanoid robot NAO. In *2012 IEEE 8th International Colloquium on Signal Processing and its Applications*, pages 188–193, 2012.

[35] Takanori Shibata, Teruaki Mitsui, Kazuyoshi Wada, Akihiro Touda, Takayuki Kumasaka, Kazumi Tagami, and Kazuo Tanie. Mental commit robot and its application to therapy of children. In *2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings (Cat. No. 01TH8556)*, volume 2, pages 1053–1058, 2001.

[36] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[37] Tomohiro Takagi and Michio Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(1):116–132, 1985.

[38] Hideki Takahashi, Kanae Matsushima, and Toshihiro Kato. The effectiveness of dance/movement therapy interventions for autism spectrum disorder: A systematic review. *American Journal of Dance Therapy*, 41(1):55–74, 2019.

[39] TU Delft Webpage. Nao image, 2020. URL `https://www.tudelft.nl/en/eemcs/current/humans-of-eemcs/nao/`. Last Time Accessed: 27-02-2021.

[40] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[41] Edwar Yazid, Matthew Garratt, and Fendy Santoso. Position control of a quadcopter drone using evolutionary algorithms-based self-tuning for first-order takagi–sugeno–kang fuzzy logic autopilots. *Applied Soft Computing*, 78:373–392, 2019.

[42] Hans-Jürgen Zimmermann. *Fuzzy set theory - and its applications*. Springer Science & Business Media, 2011.